HAMILTON and AREA PACKET NETWORK

2391 Arnold Crescent,
Burlington, Ontario,
L7P 4J2, Canada.

Hamilton and Area Packet Network
Bulletin number 2.

Well, it seems not so long ago, number 1 of the series was dropped in the mail and I settled down to a few days rest before starting on collecting goodies for this one.

This time we are going to begin a segment which I believe will be both interesting and instructive! we will select a couple of papers from the literature which contain useful information for the packet enthusiast. This issue contains a paper by da Silva and de Mercado of the DOC about packet speech, and one by Abramson reflecting on the initial ALOHA system at the University of Hawaii.

Another standard column is the bibliography!

1. I have included some quick and dirty references to the current (< 5 mos) literature on networks, packet switching and protocols. These references are meant to fill in the gaps between updates to the main bibliography.

2. The bibliography on digital communications, specifically modulation techniques, is now available (for $10.00 per copy). It contains over 400 references to modulation schemes, models, measurements etc.

3. The complete original bibliography (parts I and II) on packet, all 31 double sided pages of it, is now available for the (unfortunately increased) price of $10.00.

The suspiciously round figure of $10.00 covers the cost of duplication and mailing, and contributes a small sum to the group for further exploration as well as putting you on the mailing list.

In the circuitry department, a super simple TRAP debounce circuit by John (DVV) and his polite multiplexed modem (makes his packet wait til he finishes talking before it is sent). Connections to a TRS-80 are given by Glenn (DSP) and Max (DNM), and Robert's (EFD) 'information detector' is revealed. (Robert uses a Yaesu hand-held without accessing the squelch circuit and detects channel occupancy using audio techniques with the result that he can sneak packets thru between words on a busy channel!)

The software segment contains John's TRAP dumper routine and Glenn's TRS-80 routine to talk to the TNC.

Because there was some concern about standardization, a small segment on protocols and their role is presented along with a (proposed) standard for screen control based on the ANSI standard. An untested piece of code for the SD Sales VDB-8024 board (uses Z-80) is given as an example of how the decoding of escape sequences is accomplished.

I appologize for the bad quality on the copy of the simple 8273/S-100 bus interface schematic in the last blurb. If you really want to build one and want the schematic, send a SASE to 2391 Arnold Cres. and I will send you a better copy.

You will find a Brag sheet enclosed. If you know of anyone who would like to become a member and receive a complimentary copy of the current bulletin, have him/her fill out the form and return it to the Hamilton and Area Packet Network, c/o Stu Beal, 2391 Arnold Cres., Burlington, Ontario, L7P 4J2, Canada.

Protocols

First, a definition (related to computer technology)! Protocols are common tools designed for controlling information transfer between computer systems. The components of a protocol can include rules, message formats, frequency of interactions, error control, anticipation and flow control, accounting or synchronization and addressing strategy.

One observation is important at this point! if the system were simple enough and all users of the system had identical equipment, there would be little use for many of the protocol components! all users would merely copy the common communication program and use it. But computer systems (in particular the systems in use on the network) are anything but homogeneous, thus the requirement to identify the differences among them and take all this into account when what is called the 'Reference Architecture' is designed.

A cassette recorder and a radio can be connected together because the manufacturers agreed to standardize on miniature phone plugs as the 'Visiblity Points'

of their systems. One doesn't need to know the inner workings of the whole device, just the socket pins; the peephole into the device.

Computer systems are more complicated than this example, but the requirements for standardization are not. Even within a single system, there are requirements for protocols between the 'Layers' of software and hardware! protocols for communicating from the application program to the exec, from the exec to the drivers, from the drivers to the buss, from the buss to the interface and from the interface to the device (and possibly further). One thing to stress is that you should be able to change things within a given layer with minimal (preferably no) interference between layers (the 'Interface' between the application program and the exec shouldn't change if the driver changes (within reason)).

Once we extend our view to the network, particularly as implemented by the VADCG board, the computer terminal systems are decoupled from the net at the TIP program. There still exists an interface reference in the RS-232 or parallel link, but it doesn't impact the net in the slightest! one is free to design his own TIP and computer system.

The protocol components of 'Transmission' (RF link) and 'Network' (hop to hop validation) are already dealt with on the VADCG board as well as some rudimentary 'Flow Control' (return REJECT when buffers are full) but the application level components have yet to be provided, because all the applications have not all been identified. At the application level there is, however, a requirement for at least two components: the 'Virtual Terminal' and 'File Transfer' protocols. (The File Transfer protocol will be dealt with in another issue of the bulletin.)

What is a Virtual Terminal?

The protocol required to allow all terminals on the net to talk to each other and issue a common control language is the Reference Architecture of the Virtual terminal. First we identify all the functions of all the terminals in use. Then we do one of two things: a) standardize on the functions and mechanism of one of the terminals and make all the others emulate it or b) select a new definition and make all emulate it.

The first approach suffers in that if the chosen device doesn't have some feature that another does, the feature cannot be used at this protocol level! a higher level must be instituted to allow it's use.

The second approach is more general and will include the 'OR' of all pertinant features, but an implementation is required for all types of terminals.

What we will now propose is a standard for the Virtual Terminal. Please note all that was said about protocol rules, in particular, changes in one layer should not impact others. When you are designing software, continuously ask yourself 'If some other layer changes, how much code will I have to re-write?' and then decide on the trade off you are willing to make.

OK. The protocol for the Virtual Terminal is that selected by the American National Standards Institute. It is based on the 'Escape Sequence'.

The functioning of a terminal is composed of two aspects: communication and formatting. The keying and display of characters communicate the desired information, the arrangement of the characters on the screen (or page) is formatting. The Virtual Terminal displays information unless it has been prompted (by the Escape or some special Control character) to perform formatting. The next few pages describe the implementation of the ANSI standard on a MIME 100 terminal (Lanpar XT-100). A simple tree to examine escape sequences is then presented. This code is untested and should only be used as a guide to an implementation. It was written for a Z-80 chip, so the table look up will have to be replaced as well.

On the other side of the coin is the argument that protocol standardization can lead to unwieldy operations when conversions are done between systems. It is thus necessary to allow individual users pairs to select their own private protocols for efficiency purposes. The provision of general protocols will be necessary for the simple or once-thru operations when designing a custom protocol is not required. In the Hamilton net, some users are concerned about the virtual terminal problem and are working on it, while others are concerned with the applications and the protocols will develop out of the implementation.

This has been a brief and cursory statement about protocols. More information is available in 'A Tutorial on Protocols' by Pouzin and Zimmerman in Proc IEEE, vol 66, no.11, 1978, p1346-1370.

CONTROL CODE RESPONSE

CONTROL CHARACTER	HEX CODE	MIME 100 RESPONSE ANSI OR VT52 MODE
NUL	0H	Ignored
ENQ	05H	Transmit answerback message.
BEL	07H	Sound bell tone from keyboard.
BS	08H	Move the cursor to the left one character position, unless it is at the left margin, in which case no action occurs.
HT	09H	Move the cursor to the next tab stop, or to the right margin if no further tab stops are present on the line.
LF	0AH	This code causes a line feed or a new line operation. (See Section 1.6.3)
VT	0BH	Interpreted as LF.
FF	0CH	Interpreted as LF.
CR	0DH	Move curosr to left margin on the current line.
SO	0EH	Select G1 character set, as defined by "ESC)" sequence.
SI	0FH	Select G0 character set, as defined by "ESC(" sequence.
XON	11H	Causes terminal to resume transmission.
XOFF	13H	Causes terminal to stop transmitting all codes except XOFF and XON.
CAN	18H	If sent during an escape or control sequence, the sequence is immediately terminated and not executed. It also causes the error character to be displayed.
SUB	1AH	Interpreted as CAN.
ESC	1BH	Introduces an escape sequence.

TABLE 2.5.1

2.6 ANSI FORMAT

NOTE: A brief but restricted description of the ANSI format applicable to the MIME 100 follows. Completeness and generalization have been sacrificed in deference to expedience in discussing the standard in order to provide only the information necessary to program the MIME 100. Persons interested in the complete ANSI standard may obtain copies by writing to:

Sales Department
American National Standards Institute
1430 Broadway
New York, New York 10018

All commands begin with ESCAPE (1BH) which terminates any previously started but incomplete Command Sequence. The subsequent characters which are part of the command are not displayed. Display resumes after a complete command has been assembled. There are two basic command forms:

1. The first form consists of one or two characters other than "[" which follow ESC and uniquely define a command. Within the set of valid Escape Sequences, the value of the first character unambiguously indicates whether a subsequent final character is required to complete command, or whether the first character is the final character of the command sequence.

Examples: A) Command: NEXT LINE

ESC	E	1BH, 45H
Escape Character	Final Character	Hexidecimal Representation

B) Command: DOUBLE WIDE LINE

ESC	#	6	1BH, 23H, 36H
Escape Character	Intermediate Character	Final Character	Hexidecimal Representation

2. In the second command form, ESC is followed by "[". The character pair is called the "Control Sequence Introducer, CSI". Following the CSI are 0 or more parameters separated by semicolons and terminated by a final character. For the MIME 100, the parameters consist of decimal numbers formed by strings of 0 or more ASCII digits and the final character is alphabetic. . The parameters may be "selective" or "numeric".

SELECTIVE PARAMETERS (Ps) are used to select individual subfunctions within a selected list and must be chosen from a limited set appropriate for the command. NUMERIC PARAMETERS (Pn) explicitly represent a number and are valid within the range 0 to 250.

```
Examples!  A)  COMMAND:  Move the active (cursor) position
               to the right 15 places.

ESC [    15    C                  1BH, 5BH, 31H, 35H, 43H

 CSI     PN    Final                   Hexidecimal
             Character                 Representation

           B)  COMMAND:  Clear lamps 1-4; illuminate lamps
               1 and 3.

       Deliminiter
ESC [   0;1;3     q           1BH, 5BH, 30H, 3BH, 31H, 3BH, 33H, 71H
 CSI     Ps      Final                 Hexidecimal
               Character               Representation
```

In general, a control sequence with several selective parameters in functionally equivalent to several control sequences, each with one selective parameter. The same results as the control sequence of examble B) above could be accomplished by:

ESC[0qESC[1qESC[3q

or: ESC[;1;003q

or: ESC[qESC[1;3q

The following section describes each escape sequence valid for the MIME 100 in ANSI mode. A summary is presented in Section 2.9.1.

2.7 ANSI CONTROL SEQUENCES

CURSOR POSITION REPORT
ESC[Pn;PnR

This sequence reports the active (cursor) position of the MIME 100 to the host by means of two numeric parameters (Pn), the first specifying the line and the second specifying the column.

The numbering of lines depends on the state of the Origin Mode.

This CONTROL sequence is solicited by a Device Status Report sent from the host.

CURSOR BACKWARD
ESC[PnD

This sequence moves the active (cursor) position to the left as determined by the numeric parameter. I.e., if the value represented by the parameter value is N, the cursor is moved N spaces to the left; if zero or 1, the cursor is moved one space to the left. When attempting to move the cursor beyond the left margin, the cursor stops at the left margin.

CURSOR DOWN
ESC[PnB

This sequence moves the active (cursor) position downward without altering the column position. The distance moved is determined by the numeric parameter; i.e., if the parameter value is N, the cursor is moved N lines downward, if zero or 1, the cursor is moved one line downward. When attempting to move the cursor below the bottom margin, the cursor stops at the bottom margin.

CURSOR FORWARD
ESC[PnC

This sequence moves the active (cursor) position to the right as determined by the numeric parameter. I.e., if the parameter value is N, the cursor is moved N spaces to the right, if zero or 1, the cursor is moved one space to the right. When attempting to move the cursor to the right of the right margin, the cusor stops at the right margin.

CURSOR POSITION
ESC[Pn;PnH

This sequence moves the active (cusor) position to the position determined by two parameter values, the first specifying the line position and the second specifying the column position. A value of zero or 1 in the first parameter places the cursor in the first line; a value of zero or 1 in the second parameter places the cursor in the first column. Likewise, a value of 15;10 in the parameters will move the cursor to line 15, column 10. The default condition, with no parameters present, e.g. ESC[H, is equivalent to a Home command, and the cursor will move to the top left corner of the screen, or line 0, column 0.

The numbering of lines depends on the state of the Origin Mode.

CURSOR UP
ESC[PnA

This sequence moves the active (cursor) position upward without altering the column position. The distance moved is determined by the numeric parameter; i.e., if the parameter value is N, the cursor is moved N lines upward, if zero or 1, the cursor is moved one line upward. When attempting to move the cursor above the top margin, the cursor stops at the top margin.

DEVICE ATTRIBUTES
ESC[Pnc

1. A CONTROL sequence with either no parameters or a parameter of 0 is sent by the host to the MIME 100 requesting the MIME 100 to identify itself.

2. The MIME 100 response to this request is generated by a control sequence with the numeric parameters as follows:

OPTION PRESENT	SEQUENCE SENT
MIME 100 No Options	ESC[?1;2c
MIME 100 and STP	ESC[?1;3c
MIME 100 and GPO	ESC[?1;6c
MIME 100, GPO, and STP	ESC[?1;7c
GPO - Graphics Option	
STP - Processor Option	

DEVICE STATUS REPORT
ESC[PsR

The general status of the MIME 100 is requested and reported according to the following selective parameter (Ps):

PARAMETER	PARAMETER MEANING
0	Response from MIME 100—Ready, No malfunctions detected (default)
3	Response from MIME 100—Malfunction—retry
5	Command from host—Please report status (using a DSR control sequence)

A DSR with a parameter value of 0 or 3 is always sent as a response to a DSR from the host with a parameter value of 5.

6 (using a CPR control sequence)	Command from host—Please report active position

ERASE IN DISPLAY
ESC[PsJ

This sequence erases some or all of the characters in the display as determined by the parameter. Any complete line erased by this sequence will return that line to single width mode.

PARAMETER	PARAMETER MEANING
0	Erase from cursor position to the end of the screen (default)
1	Erase from the start of the screen to and including the cursor position
2	Erase all of the display—all the lines are erased and changed to single-width. The cursor does not move.

ERASE IN LINE
ESC[PsK

This sequence erases some or all of the characters in the active line as determined by the parameter.

PARAMETER	PARAMETER MEANING
0	Erase from the cursor position to the end of the line, inclusive (default)
1	Erase from the start of the line to and including the cursor position
2	Erase all of the line, inclusive

HORIZONTAL TABULATION SET
ESC H

This sequence sets one horizontal stop at the cursor position.

HORIZONTAL AND VERTICAL POSITION
ESC[Pn;Pnf

This sequence moves the active (cusor) position to the position determined by two parameter values, the first specifying the line position and the second specifying the column position. A value of zero or 1 in the first parameter places the cursor in the first line; a value of zero or 1 in the second parameter places the cursor in the first column. Likewise, a value of 15;10 in the parameters will move the cursor to line 15, column 10. The default condition, with no parameters present, e.g. ESC[f, moves the cursor to the home position. The numbering of lines and columns depends on the reset or set state of the Origin Mode.

INDEX
ESC D

This sequence causes the cursor to move downward one line without changing the column position. If the cursor is at the bottom margin, a scroll up is performed.

LINE FEED/NEW LINE MODE
Set Mode:ESC[20h
Reset Mode:ESC[201

The Reset Mode control sequence, e.g. ESC[201, causes the cursor to move vertically, and the RETURN key (CR) to send the single code CR. The Set Mode control sequence, e.g. ESC[20h, causes the cursor to move to the first position of the following line, and causes the RETURN key to send the two codes (CR,LF). The state of this option upon power up is determined by a "switch" setting. See Section 1.6.3.

NEXT LINE
ESC E

This sequence causes the active position to move to the first position on the next line downward. If the active position is at the bottom margin, a scroll up is performed.

SCREEN ALIGNMENT DISPLAY
ESC#8

The command fills the entire screen area with uppercase E's for screen focus and alignment. This command is used by Micro-Term manufactuing personnel.

ANSI/VT52 MODE
Set Mode: N/A
Reset Mode: ESC[?2l

The Reset Mode control sequence, e.g., ESC[?2l, causes only VT52 compatible escape sequences to be interpreted and executed. The Set Mode control sequence causes only ANSI "compatible" escape and control sequences to be interpreted and executed.

AUTO REPEAT MODE
Set Mode: ESC[?8h
Reset Mode: ESC[?8l

The Set Mode control sequences cause certain keyboard keys to auto-repeat. The Reset Mode control sequences cause no keyboard keys to auto-repeat.

AUTOWRAP MODE
Set Mode: ESC[?7h
Reset Mode: ESC[?7l

The Set Mode control sequences, e.g., ESC[?7h, cause any displayable characters received when the cursor is at the right margin to advance to the start of the next line, performing a scroll up if required and permitted. The Reset Mode control sequence, e.g., ESC[?7l, cause these characters to overwrite any previous characters at the right margin.

CURSOR KEYS MODE
Set Mode: ESC[?1h
Reset Mode: ESC[?1l

This mode is only effective when the terminal is in keypad application mode and the ANSI/VT52 mode is set. Under these conditions, the Set Mode control sequences will cause the four cursor function keys to send application functions. The Reset Mode control sequences will cause the four cursor function keys to send ANSI cursor control commands. See Section 2.1.5.

COLUMN MODE
Set Mode: ESC[?3h
Reset Mode: ESC[?3l

The Set Mode control sequences will cause a maximum of 132 columns on the screen. The Reset Mode control sequences will cause a maximum of 80 columns on the screen.

DOUBLE HEIGHT LINE
Top Half: ESC#3
Bottom Half: ESC#4

The line containing the cursor becomes the top or bottom half of a double-height, double-width line, as determined by the control sequences. In order to insure full double-height characters, the control sequences must be used in pairs on adjacent lines, and the same character output must be sent to both lines. If the line was single-width, single-height, all characters to the right of the center of the screen are lost. If the cursor is located to the left of the center screen, it will remain in the same character position; if the cursor is to the right of the center screen, it is moved to the right margin.

DOUBLE-WIDTH LINE
ESC#6

This sequence causes the line that contains the cursor to become double-width, single-height. If the line was single-width, single-height, all characters to the right of the center of the screen are lost. If the cursor is located to the left of the center screen, it will remain in the same character position; if the cursor is to the right of the center screen, it is moved to the right margin.

IDENTIFY TERMINAL
ESC Z

This sequence causes the same response as the ANSI DEVICE ATTRIBUTES. This sequence will not be supported in future terminals, therefore the ANSI device attributes command sequence should be used by any new software.

INTERLACE MODE
Set Mode: ESC[?9h
Reset Mode: ESC[?9l

The Set Mode control sequence, e.g. ESC[?9h, (interlace) causes the video processor to display 480 scan lines per frameif the graphics processor option

has been installed. The Reset Mode control sequences, e.g., ESC[?91 (non interlace) causes the video processor to display 240 scan lines per frame. There is no increase in character resolution.

KEYPAD APPLICATION MODE
ESC=

The auxiliary keypad keys and cursor control keys will transmit control sequences as defined in Tables 2.1.5 and 2.1.6.

KEYPAD NUMERIC MODE
ESC >

The auxiliary keypad keys will send ASCII codes corresponding to the characters on the keys. The cursor control keys will send cursor controls.

LOAD LEDS
ESC[Psq

Load the four programmable LED's on the keyboard according to the selective parameter(s).

PARAMETER	PARAMETER MEANING
0	Clear LED's L1 through L4
1	Light L1
2	Light L2
3	Light L3
4	Light L4

The LED numbers are indicated on the keyboard.

ORIGIN MODE Set Mode: ESC[?6h Reset Mode: ESC[?6l

The Set Mode control sequence, e.g., ESC[?6h, causes the origin (home) to be at the upper left character position within the margins. Refer to SET TOP AND BOTTOM MARGINS. Once margins have been defined, the line and column numbers are relative to those margin settings. The cursor is not allowed to be positioned outside the margins.

The Reset Mode control sequence, e.g., ESC[?6l, causes the origin (home) to be at the upper-left character position on the screen (column 1, line 1). If margins have been set (refer to SET TOP AND BOTTOM MARGINS), the line and column numbers are independent of these settings. The cursor may be positioned outside the margins with a cursor position or horizontal and vertical position control.

The cursor is moved to the new home position when this mode is set or reset.

Lines and columns are numbered consecutively, with the origin (home) being line 1, column 1 (upper left character position).

RESTORE CURSOR
ESC 8

This sequence causes the previously saved cursor position, graphic rendition, and character set to be restored. Refer to SAVE CURSOR.

SAVE CURSOR
ESC 7

This sequence causes the cursor position, graphic rendition, and character set to be saved. Refer to RESTORE CURSOR.

SCROLLING MODE
Set Mode: ESC[?4h
Reset Mode: ESC[?4l

The Set Mode control sequence, e.g., ESC]?5h, causes the scroll to be "smooth" at a maximum rate of six lines per second. The Reset Mode control sequence, e.g. ESC[?5l, causes the scroll to "jump" instantaneously.

SCREEN MODE
Set Mode: ESC[?5h
Reset Mode: ESC[?5l

The Set Mode control sequence, e.g., ESC[?5h, causes the screen to be white with black characters. The Reset Mode, e.g., ESC[?5l, causes the screen to be black with white characters.

SET TOP AND BOTTOM MARGINS
ESC[Pn;Pnr

This control sequence lets the top and bottom margins to define the scrolling region. The first numeric parameter in the control sequence refers to the first line in the scrolling region; the second numeric parameter refers to the bottom line in the scrolling region. Default is the entire screen, i.e., no margins—the entire screen will scroll. The minimum size of the scrolling region allowed is two lines, i.e., the top margin (line number) must be less than the bottom margin (line number). The cursor is placed in the home position. Refer to ORIGIN MODE.

SINGLE-WIDTH LINE
ESC#5

This control sequence causes the line containing the cursor to become single-width, single-height. The cursor remains at the same character position. This is the default condition for all new lines on the screen.

REVERSE INDEX
ESC M

This control sequence causes the cursor to move to the same horizontal position on the preceding line. If the cursor position is at the top margin, a scroll down is performed.

RESET TO INITIAL STATE
ESC c

This control sequence resets the MIME 100 to its initial state, i.e. the state it has after it is powered on. This sequence also causes the execution of the power-up self-test.

RESET MODE
ESC[Ps;Ps;...;Psl

This control sequence resets one or more MIME 100 modes as defined by each selective parameter in the parameter string. Each mode to be reset is specified by a separate parameter. Refer to SET MODE.

SET MODE
ESC[Ps;...;Psh

This control sequence sets one or more MIME 100 modes as defined by each selective parameter in the parameter string. Each mode to be set is specified by a separate parameter. A mode is considered set until it is reset by a Reset Mode control sequence. Refer to RESET MODE.

PARAMETER	REFER TO MODE FUNCTION	"SET"	"RESET"
1	Cursor Key	Application	Cursor
2	ANSI/VT52	ANSI	VT52
3	Column	132	80
4	Scrolling	Smooth	Jump
5	Screen	Reverse	Nomal
6	Origin	Relative	Absolute
7	Auto Wrap	Enabled	Disabled
8	Auto Repeat	Enabled	Disabled
20	New Line	New Line	Line Feed

All other parameters are ignored.

SELECT CHARACTER SET

The appropriate G0 and G1 character sets are designated from one of the five possible character sets. The G0 character set is enabled by the code SI (Shift In), while the G1 character set is enabled by the control code SO (Shift Out).

G0 SETS SEQUENCE	G1 SETS SEQUENCE	MEANING
ESC(A	ESC)A	United Kingdom Set
ESC(B	ESC)B	ASCII Set
ESC(0	ESC)0	Special Graphics
ESC(1	ESC)1	Alternate Character ROM Standard Character Set
ESC(2	ESC)2	Alternate Character ROM Special Graphics

The United Kingdom and ASCII sets conform to the "ISO international register of character sets to be used with escape sequences". The other sets are private character sets. Spcial Graphics means that the graphic characters for the codes 6F Hex to 7E Hex are replaced with other characters (see Table 2.3.1). When a specified character set is enabled through the use of the Select Character Set codes SI or SO, that character set will be used until another Select Character Code (SI or SO) is received.

SELECT GRAPHIC RENDITION
ESC[Ps;...Psm

This control sequence enables the graphic rendition as specified by the parameter(s). All characters transmitted to the MIME 100 following the control sequence are rendered according to the specified parameter(s) until the next occurance of a Select Graphic Rendition control sequence.

PARAMETER	PARAMETER MEANING
0	Attribute Off (Default)
1	Bold or Increased intensity
4	Underscore
5	Blink
7	Negative (reverse) image

All other parameter values are ignored.

TABULATION CLEAR
ESC[Psg

This control code sequence causes tabs to be cleared as specified by the numeric parameter. Default value is 0.

PARAMETER	PARAMETER MEANING
0	Clear the horizontal tab stop at the active position.
3	Clear all horizontal tab stops.

All other parameter values are ignored.

2.9 CONTROL SEQUENCE SUMMARY

2.9.1 ANSI MODE CONTROL SEQUENCES

CURSOR MOVEMENT

Cursor Up	ESC[PnA	
Cursor Down	ESC[PnB	
Cursor Forward (right)	ESC[PnC	
Cursor Backward (left)	ESC[PnD	
Direct Cursor Addressing	ESC[Pl;PcH	Pl=line #
or	ESC[Pl;Pcf	Pc=column #
Index	ESC D	
New Line	ESC E	
Reverse Index	ESC M	
Save Cursor and Attributes	ESC 7	
Restore Cursor and Attributes	ESC 8	

ERASING

From Cursor to End of Line	ESC[K or ESC[0K
From Beginning of Line to Cursor	ESC[1K
Entire Line Containing Cursor	ESC[2K
From Cursor to End of Screen	ESC[J or ESC[0J
From Beginning of Screen to Cursor	ESC[1J
Entire Screen	ESC[2J

CHARACTER LINE SIZE

Change this line to double-height top half	ESC#3
Change this line to double-height bottom half	ESC#4
Change this line to single-width single-height	ESC#5
Change this line to double-width single-height	ESC#6

CHARACTER ATTRIBUTES

ESC[Ps;Ps;Ps;....;Ps m

Ps refers to a selective parameter. Multiple parameters are separated by the semicolon character (3BH). The parameters are executed in order and have the following meanings:

0 or None	All Attributes Off
1	Bold on
4	Underscore On
5	Blink On
7	Reverse Video On

CHARACTER SETS

CHARACTER SET	G0 DESIGNATOR	G1 DESIGNATOR
United Kingdom (UK)	ESC(A	ESC)A
United States (USASCII)	ESC(B	ESC)B
Special Graphics Characters and Line Drawing Set	ESC(0	ESC)0
Alternate Character ROM	ESC(1	ESC)1
Alternate Character ROM Special Graphics Characters	ESC(2	ESC)2

SCROLLING REGION

ESC[Pt;Pb r

Pt is the number of the top line of the scrolling region; Pb is the number of the bottom line of the scrolling region and must be greater than Pt.

TAB STOPS:

Set tab at current column	ESC H
Clear tab at current column	ESC[g or ESC[0g
Clear all tabs	ESC[3g

PROGRAMMABLE LEDS

ESC[Ps;Ps;...Ps q

Ps are selective parameters separated by semicolons (3BH) and executed in order, as follows:

0 or None	All LEDs Off
1	L1 On
2	L2 On
3	L3 On
4	L4 On

MODES

	TO SET		TO RESET	
MODE NAME	MODE	SEQUENCE	MODE	SEQUENCE
Line feed/new line	NEW line	ESC[20h	Line feed	ESC[20l
Cursor key mode	Application	ESC[?1h	Cursor	ESC[?1l
ANSI/VT52 mode	ANSI	N/A	VT52	ESC[?2l
Column mode	132 Col	ESC[?3h	80 Col	ESC[?2l
Scrolling mode	Smooth	ESC[?4h	Jump	ESC[?4l
Screen mode	Reverse	ESC[?5h	Normal	ESC[?5l
Origin mode	Relative	ESC[?6h	Absolute	ESC[?6l
Wraparound	On	ESC[?7h	Off	ESC[?7l
Auto repeat	On	ESC[?8h	Off	ESC[?8l
Keypad mode	Application	ESC =	Numeric	ESC[?>

REPORTS

Cursor Position Report

Invoked by	ESC[6n	Pline;Pcol
Response is	ESC[Pl;PcR	

Status Report

Invoked by	ESC[5n	
Response is	ESC[0n	(terminal OK)
	ESC[3n	(terminal not OK)

What Are You

Invoked by	ESC[c or
	ESC[0
Response is	ESC[?1;Ps c

Ps is the "option present" parameter with the following meaning:

Ps	Meaning
2	MIME 100, no graphics
3	Processor option (STP)
6	Graphics Processor Option (GPO)
7	STP and GPO

RESET

Reset causes the power-up reset routine to be executed.

ESC c

CONFIDENCE TESTS

Fill Screen with "Es"	ESC#8
Invoke Tests	ESC[2;9y

Tests exercise ROM. RAM, and EAROM. Tests are repeated indefinitely until failure or power off.

The Hamilton and Area Packet Network

The second meeting of the Hamilton and Area Packet Network will be held at:

Seminar room,
The Canada Centre for Inland Waters,
867 Lakeshore Road,
Burlington, Ontario.

on Sunday, March 8th at 13:00 EST.

Subjects to be thrashed out include Applications, Protocols and ...

Plan to attend and if you have any friends who are really interested in getting into Packet operations, feel free to invite them.

Directions from Toronto:
QEW towards Niagara at Burlington,
exit to the Beach Blvd./Lakeshore Road
just before the Skyway bridge,
Right at stoplight on Lakeshore Road,.
about 1/2 mile on the right, under the
Skyway is C.C.I.W.

From the South,
QEW to Beach Blvd.,
Left at light onto Beach Blvd,
Cross the Lift Bridge at the Canal,
Left at Main entrance into C.C.I.W.,
just past the bridge.

```
                ;       ESCAPE SEQUENCE PROCESSING
                ;
                ;       THE ESCAPE SEQUENCE DECODING IS HANDLED BY A TREE OF
                ;       POSSIBLE RESPONSES. TO RIGHT IS SUCCESS, DOWN, FAILURE:
                ;
                ;       ESC --> [ --> PN --> ; --> PN --> H --> ; CURSOR DIRECT
                ;       !       !     !
                ;       !       !     !    A -->  ;CURSOR UP
                ;       !       !     !    B -->  ;CURSOR DOWN
                ;       !       !     !    C -->  ;CURSOR FORWARD (RIGHT)
                ;       !       !     !    D -->  ;CURSOR BACKWARD (LEFT)
                ;       !       !     !    J -->  ;SCREEN ERASE DEPENDING ON PN
                ;       !       !     !    K -->  ;LINE ERASE DEPENDING ON PN
                ;       !       !     !    M -->  ;ATTRIBUTES DEPENDING ON PN
                ;       !       !     !
                ;       !       !     !    ABORT (NONE OF ABOVE)
                ;       !       !     !
                ;       !       !     H -->  ;HOME CURSOR
                ;       !       !     !
                ;       !       !     ABORT (NONE OF ABOVE)
                ;       !       !
                ;       !       ( -->  B -->  ;STANDARD ASCII CHARACTER SET
                ;       !       !      2 -->  ;ALTERNATE SET WHEN SELECTED BY SO
                ;       !       !      !
                ;       !       !      ABORT
                ;       !       !
                ;       !       D -->  ;CURSOR DOWN A LINE AND SCROLL
                ;       !       E -->  ;DOWN LINE, COLUMN 1 AND SCROLL IF NEEDED
                ;       !       M -->  ;CURSOR UP A LINE AND SCROLL IF NEEDED
                ;       !       7 -->  ;SAVE CURSOR AND ATTRIBUTES
                ;       !       8 -->  ;RESTORE CURSOR AND ATTRIBUTES
                ;       !       C -->  ;RESET DISPLAY SYSTEM
                ;       !
                ;       DISPLAY NON ESCAPE SEQUENCE

00C7 CD0001     ESCAPE: CALL    NEXTCH          ;GET NEXT CHARACTER FROM HOST
00CA FE5B               CPI     HLBRAK          ;IS IT '['
                        JRNZ    TEST2           ;NO, TRY NEXT POSSIBILITY
00CC+201D               DB      20H,TEST2-$-1
                ;
                ;       ESC [  FOUND, GET PARAMETERS TIL NON ;
                ;
                ;       CLEAR OUT PREVIOUS SETTINGS OF 'PARAMETERS'

00CE CDE601     NUMIN:  CALL    GETPAR          ;GET A NUMERIC PARAMETER FROM HOST
00D1 70                 MOV     M,B             ;POKE INTO MEMORY
00D2 FE3B               CPI     HSEMI           ;IS TERMINATOR A ';'
00D4 23                 INX     H
                        JRZ     NUMIN           ;YES, TRY FOR ANOTHER PARAMATER
00D5+28F7               DB      28H,NUMIN-$-1
                ;
                ;       TEST FOR 'ESC [ P1 ; ...PN ' FOLLOWED BY :
                ;
00D7 010800             LXI     B,TBL2L         ;FETCH LENGTH OF THIS TABLE
00DA 211C02             LXI     H,TBL2C         ;ADDRESS OF CHARACTER TABLE
                        CCIR                    ;SEARCH FOR THE MATCH
00DD+EDB1               DB      0EDH,0B1H
00DF 213402             LXI     H,TBL2D         ;GET END OF DISPATCH TABLE
                        DSBC    BC              ;SUBTRACT REMAINING COUNT
00E2+ED42               DB      0EDH,BC$8+42H
                        DSBC    BC              ;    TWICE
00E4+ED42               DB      0EDH,BC$8+42H
00E6 E9                 PCHL                    ;AND GO PROCESS
                        JR      QUIT            ;AT END ORIF NONE OF ABOVE, TERMINATE
00E7+181E               DB      18H,QUIT-$-1
```

```
                ;
                ;       TEST FOR 'ESC (' FOLLOWED
                ;
00E9 FE28       TEST2:  CPI     HLPAR           ;'('?
                        JRNZ    TEST3           ;NO, TRY OTHER STUFF
00EB+200A               DB      20H,TEST3-$-1
00ED FE42               CPI     HB              ;'B'?
00EF CA0000             JZ      STDCHR          ;YES, SET STANDARD CHARACTER
00F2 FE32               CPI     H2              ;'2'?
00F4 CA0000             JZ      ALTCHR          ;YES, ALTERNATE CHARACTER SET
                ;
00F7 010700     TEST3:  LXI     B,TBL3L         ;FETCH LENGTH OF THIS TABLE
00FA 213602             LXI     H,TBL3C         ;ADDRESS OF CHARACTER TABLE
                        CCIR                    ;SEARCH FOR THE MATCH
00FD+EDB1               DB      0EDH,0B1H
00FF 214802             LXI     H,TBL3D         ;GET END OF DISPATCH TABLE
                        DSBC    BC              ;SUBTRACT REMAINING COUNT
0102+ED42               DB      0EDH,BC$8+42H
                        DSBC    BC              ;    TWICE
0104+ED42               DB      0EDH,BC$8+42H
0106 E9                 PCHL                    ;AND GO PROCESS
                ;
                QUIT:
                ;
                ;       VARIOUS ROUTINES CALLED BY PROCESSES
                ;
                ;       FETCH THE NEXT CHARACTER FROM THE HOST
                ;
01DD 37         NEXTCH: STC
01DE 3F                 CMC                     ;RETURN HERE ON INTERRUPT
01DF FB                 EI
01E0 76                 HLT
01E1 DB20               IN      CRTIN           ;FETCH THE CHARACTER
                        RES     7,A             ;TRIM OFF THE TOP BIT
01E3+CBBF               DB      0CBH,7$8+A+80H
01E5 C9                 RET                     ;RETURN IT TO THE CALLER
                ;
                ;       FETCH NUMBERIC CHARACTERS AND FORM A NUMERIC PARAMETE
                ;       STOP ON NON-NUMERIC (; OR OTHER)
                ;
01E6 0600       GETPAR: MVI     B,0             ;INITIALIZE THE RESULT TO 0
01E8 CDDD01     GETNXT: CALL    NEXTCH          ;GET THE NEXT BYTE IN
01EB CD0000             CALL    ISDIGIT         ;CHECK IF DIGIT
01EE D8                 RC                      ;NO, RETURN WITH PARAMETER IN
                                                ;AND TERMINATOR IN A
01EF CD0000             CALL    MUL10           ;MULTIPLY # BY 10
01F2 80                 ADD     B               ;FORM THE CURRENT TOTAL
01F3 47                 MOV     B,A             ;MOVE BACK TO B
                        JR      GETNXT          ;TRY FOR ANOTHER DIGIT
01F4+18F2               DB      18H,GETNXT-$-1
                ;
                ;       TABLE 1 IS THE DISPATCH TABLE FOR CONTROL CHARACTERS
                ;
01F6 0A         TBL1C:  DB      HLF             ;IS IT LF?
01F7 0D                 DB      HCR             ;IS IT CR?
01F8 11                 DB      HDC1            ;SET CONTROL BIT #1?
01F9 12                 DB      HDC2            ;CLEAR CONTROL BIT #1?
01FA 13                 DB      HDC3            ;SET CONTROL BIT #2?
01FB 14                 DB      HDC4            ;CLEAR CONTROL BIT #2?
01FC 08                 DB      HBS             ;BACKSPACE?
01FD 09                 DB      HTAB            ;TAB?
01FE 0E                 DB      HSO             ;SHIFT OUT (ALTERNATE CHARACT SET
01FF 0F                 DB      HSI             ;SHIFT IN (NORMAL CHARACTER S
0200 0C                 DB      HFF             ;CLEAR PAGE?
0201 1B                 DB      HESC            ;ESC? (CURSOR MOTION)
000C =          TBL1L   EQU     $-TBL1C
```

```
                ;
                ;       DISPATCH TABLE FOR CONTROL CHARACTERS
                ;
0202 2B01               DW      LF              ;YES
0204 4601               DW      CR
0206 C101               DW      SC1
0208 C801               DW      RC1
020A CF01               DW      SC2
020C D601               DW      RC2
020E 5001               DW      CTRH
0210 9901               DW      TAB
0212 A901               DW      SHIFTO
0214 A801               DW      SHIFTI
0216 5A01               DW      CLEAR
0218 C700               DW      ESCAPE
021A 5600       TBL1D:  DW      SHOW            ;TEST ENDS IN FAILURE
                ;
                ;       TABLE FOR 'ESC [ PN... ' FOLLOWED BY
                ;
021C 41         TBL2C:  DB      HA              ;'A'?
021D 42                 DB      HB              ;'B'?
021E 43                 DB      HC              ;'C'?
021F 44                 DB      HD              ;'D'?
0220 48                 DB      HH              ;'H'?
0221 4A                 DB      HJ              ;'J'?
0222 4B                 DB      HK              ;'K'?
0223 4D                 DB      HM              ;'M'?
0008 =          TBL2L   EQU     $-TBL2C
                ;
                ;       DISPATCH TABLE FOR 'ESC ['
                ;
0224 0000               DW      CURSUP          ;CURSOR UP WITHOUT SCROLL
0226 0000               DW      CURSDN          ;CURSOR DOWN WITHOUT SCROLL
0228 0000               DW      CURSRI          ;CURSOR RIGHT STOP AT MARGIN
022A 0000               DW      CURSLF          ;CURSOR LEFT, STOP AT MARGIN
022C 8201               DW      PLACE           ;DIRECT CURSOR ADDRESS (INCLUDES HOME)
022E 0000               DW      ERASCR          ;ERASE SCREEN DEPENDING ON P1
0230 0000               DW      ERALIN          ;ERASE LINE DEPENDING ON P1
0232 1000               DW      AUXCTL          ;MODIFY ATTRIBUTES
0234 0701       TBL2D:  DW      QUIT
                ;
                ;       ESCAPE BUT NOT [
                ;
0236 28         TBL3C:  DB      HLPAR           ;'('?
0237 44                 DB      HD              ;'D'?
0238 45                 DB      HE              ;'E'?
0239 4D                 DB      HM              ;'M'?
023A 37                 DB      H7              ;'7'?
023B 38                 DB      H8              ;'8'?
023C 63                 DB      HLC             ;'C'?
0007 =          TBL3L   EQU     $-TBL3C
                ;
023D F700               DW      TEST3           ;DO PROCESS FOLLOWER
023F 0000               DW      CURSDS          ;CURSOR DOWN AND SCROLL IF NESS
0241 0000               DW      CRLFS           ;CR + LF + SCROLL IF NESS
0243 0000               DW      CURSUS          ;CURSOR UP AND SCROLL IF NESS
0245 0000               DW      SAVE            ;SAVE CURSOR LOC AND ATTRIBUTES
0247 0000               DW      RESTOR          ;RESTORE CURSOR LOC AND ATTRIBUTES
0249 AA00               DW      RESET           ;RESET DISPLAY SYSTEM
024B 0701       TBL3D:  DW      QUIT
```

This bibliography was made by searching the last 6 months of Current Contents, Engineering, Technology and Applied Sciences for the words ALOHA, Network, Packet or Protocol. The number after the sequence number is the issue number of CC the citation was found in.

Certain liberties have been taken in leaving out citations found! if the language was Russian, if the network was obviously a TV or Radio broadcast network or if it was a movie titled 'Aloha, Bobby and Rose'.

1. (79-50)
 -several papers in the PERSONAL COMPUTING DIGEST, 1980 NCC Personal Computing festival, Anaheim, CA, may 20-22 1980. AFIPS
 -

2. (79-50)
 -A new class of protocols for multiple access in satellite networks,
 -IEEE Transactions on Automatic control, V25 no5, Oct 80 p 865

3. (79-50)
 -Communications protocol for dumb terminals,
 -Computer communications, IPC Science and Technology press, ltd. V3, no5 oct 80 p 224

4. (79-49)
 -Teleinformatics 79
 -Proc. of Teleinformatics '79, Paris, France.
 message systems, teleconferencing, technical issues, case studies.

5. (79-48)
 -the Space informatics network experiment
 -ESA bulletin, V4 no3 1980 p 247

6. (79-48)
 -several papers! Special issue on military communications, Part 1 of 2 parts
 -IEEE transactions on Communications, v 28 no 9 part 1 sept 1980

7. (79-48)
 -part 2 of previous item
 -

8. (79-47)
 -International Conference on Radio spectrum conservation techniques,
 -July 7-9 1980, London England

9. (79-47)
 -The critical satellite technical issues of future pervasive broadband low-cost communication networks
 -ACTA ASTRONAUTICA, v7 no10, oct 1980

10. (79-46)
 -Modelling a packet switched network
 -Computer communications v3 no4 aug 1980

11. (79-46)
 -Cernet Packet routing method
 -Computer communications v3 no4 aug 1980

12. (79-45)
 -1980 International Zurich seminar on Digital Communications,
 -Digital communications in Wireless systems available thru IEEE new york. 298 pages IEEE cat no 80-Ch1521-4Com Lots of interesting stuff (I had a look) Microwave applications seem to be prime but otherwise useful. sb

13. (79-44)
 -Sixth Data Communications Symposium, Pacific Grove, CA Nov 27-29, 1979 (ACM IEEE)
 -IEEE New York, cat no 79CH 1405-0 CSCB/C $20.00
 Communication based data bases, Protocols, design, data voice and image integration, EFT, etc.

14. (79-44)
 -Throughput of unslotted ALOHA channels with arbitrary packet interarrival time distributions
 -IEEE transactions on Communications V28 no8 part II aug 1980 p1422

15. (79-51)
 -COMPUTATIONAL ALGORITHMS FOR PRODUCT FORM QUEUEING NETWORKS
 -COMM ACM V 23 N 10 573

16. (79-51)
 -A HIERARCHICAL ROUTING AND FLOW CONTROL POLICY (HRFC) FOR PACKET SWITCHED NETWORKS
 -IEEE TRANS ON COMPUT. V 29 N 11 971

17. (79-51)
 -A QUEUEING THEORY BASED ANALYLYTIC MODEL OF A DISTRIBUTED COMPUTER NETWORK
 -IEEE TRANS COMPUT V 29 N 11 994

18. (79-52)
 -ENUMERATION OF ALL SIMPLE PATHS IN A COMMUNICATIONS NETWORK
 -MICROELECTRONICS AND RELIABILITY V 20 N 4 419

19. (80-1)
 -ON THE THRUPUT OF AN ALOHA CHANNEL WITH VARIABLE LENGTH PACKETS
 -IEEE TRANS ON COMMUN. V 28 N 11 1932

20. (80-1)
 -ON THE STABILITY OF SLOTTED ALOHA SYSTEMS
 -IEEE TRNAS ON COMMUN, V 28 N 11 1936

21. (80-1)
 -ADPTIVE CONTROL OF MESSAGE FLOW DISTRIBUTION IN COMMUNICATION NETWORKS
 -ENGINEERING CYBERNETICS V 17 N 5 P 103

22. (80-1)
 -ISSUE ON! PERSPECTIVES ON LIBRARY NETWORKS AND RESOURCE SHARING
 -J AM SOC INFO SCIENCE V 31 N 6

23. (80-2)
 -A NEW PROTOCOL FOR A DISTRIBUTED LOOP COMMUNICATION NETWORK
 -IN INFO II: 2ND INT CONF ON INFORMATION SCIENCES AND SYSTEMS, UNIV. PATRAS, PATRAS, GREECE, JULY 1979 P 297

24. (80-2)
 -DISTRIBUTED PROCESSING NETWORK SURVIVABILITY, 1.
 -INFO II (OP.CIT) P 313

25. (80-2)
 -QUEUEING ANALYSIS OF RANDOM ACCESS PROTOCOLS
 -INFO II P 333

26. (80-2)
-CONGESTION AVOIDANCE IN HIERARCHICAL PACKET SWITCHED NETWORKS
-INFO II P 357

27. (80-2)
-CONGESTION CONTROL IN PACKET NETWORKS
-INFO II P 367

28. (80-2)
-CONGESTION CONTROL TECHNIQUES FOR PACKET NETWORKS
-INFO II P 377

29. (80-2)
-ROUTING AND FLOW CONTROL IN VIRTUAL CIRCUIT NETWORKS
-INFO II P 386

30. (80-2)
-TOWARDS A UNIFIED DEFINITION OF TRAFIC CONTROL IN COMMUNICATION NETWORKS
-INFO II P 395

31. (80-2)
-VIDEOTEX NETWORKS
-COMPUTER V 13 N 12 P 8

32. (80-2)
-A STUDY OF TIMESHARING NETWORKS CONSIDERED AS QUEUEING NETWORKS OF EXPONENTIAL SERVERS
-THE COMPUTER JOURNAL V 23 N 4 P 290

33. (80-4)
-PERFORMANCE EVALUATION OF A SATELLITE LINKED EXPERIMENTAL NETWORK
-IEEE TRANS ON AEROSPACE AND ELECTRONIC SYSTEMSV 16 N 6 P 771

34. (80-4)
-THE METHOD OF EQUIVALENT SUBSTITUTIONS FOR CALCULATING FRAGMENTS OF COMMUNICATION NETWORKS FOR A DIGITAL COMPUTER, I
-ENGINEERING CYBERNETICS V 17 N 6 P 66

35. (80-4)
-HIGH CAPACITY - LOW DELAY PACKET BROADCAST MULTI-ACCESS
-IEEE TRANS ON AEROSPACE AND ELECTRONIC SYSTS V 16 N 6 P 830

36. (80-3)
-NBS NETWORK MEASUREMENT METHODOLOGY AS APPLIED TO SYNCHRONOUS COMMUNICATIONS
-NBS SPECIAL PUBLICATION no500-65, 16TH MEETING OF THE COMPUTER PERFOR. EVAL. GROUP, ORLANDO FLA., OCT 80

37. (80-3)
-SPECIAL FEATURE! COMMUNICATION! THE WAY AHEAD, BPO P-S NETWORK, SPEECH BAND MODEMS DIGITAL CROSSATALK MEAS. ETC.
-ELECTRONICS AND POWER, V 26 N 9 P 719 - 750

38. 43
-SEVERAL PAPERS IN CONF ON COMMUNICATIONS EQUIPMENT AND SYSTEMS, BIRMINGHAM ENGLAND APR 16-18 80 BY IEE
-

39. 43
-SEVERAL PAPERS IN 2ND WORKING CONF ON COMMAND LANGUAGES , BERCHTESGADEN, F.R.G. SEPT 79
-

40. 41
-PROSPECTS OF DATA TRANSMISSION NETWORKS
-COMMUTATION AND TRANSMISSION V 2 N 2 P 7

41. 40
-EVOLUTION TOWARD SYNCHRONIZINZ DIGITAL NETWORK IN JAPAN
-JAPAN TELECOMMUNICATIONS REWIEW, V 22 N 3 P 223

42. 40
-PARAMETERS AND COMMUNICATION EFFICIENCIES IN THE MODELLING OF A PACKET SWITCHING NETWORK
ELECTRICAL COMMUNICATION, V 55 N 1 P 46

43. 40
-DEMAND ASSIGNMENT MULTIPLE ACCESS SYSTEM FOR THIN ROUTE NETWPRKS
-ELECTRICAL COMMUNICATION V 55 N 1 P 70

44. 40
-A DECENTRALIZED ALGORITHM FOR OPTIMAL ROUTING IN DATA-COMMUNICATION NETWORKS
-LARGE SCALE SYSTEMS (THEORY AND APPLICATIONS) V 1 N 3 P 149

45. 40
-DIVIDE AND CONQUER - A COMPUTER NETWORKING PREMISE
-26TH INT INSTRUMENTATION SYMP, SEATTLE WA MAY 5-8 80

46. 39
-A NOT SO EUPHORIC VIEW OF GEE WHIZ NETWORKING
-INFOSYSTEMS V 27 N 8 P 50

47. 39
-ON A CLASS OF MULTISTAGE INTERCONNECTION NETWORKS
-IEEE TRANS ON COMPUT, V 29 N 8 P 694

48. 39
-CONCERNING MEAN PACKET DELAY AS A PERFORMANCE MEASURE IN SINGLE CHANNEL NETWORKS
-PROC IEEE V 68 N 8 P 1022

49. 37
-ANALYSIS OF SHARED FINITE STORAGE IN A COMPUTER NETWORK NODE ENVIRONMENT UNDER GENERAL TRAFFIC CONDITIONS
-IEEE TRANS ON COMMUN V 28 N 7 P 992

50. 37
-A QUEUEING ANALYSIS OF 2/ARQ PROTOCOLS
-IEEE TRANS ON COMMUN V 28 N 7 P 1004

51. 37
-PACKET SWITCHING IN RADIO CHANNELS
- NEW CONFLICT FREE MULTIPLE ACCESS SCHEMES
-IEEE TRANS ON COMMUN V 28 N 7 P 1015

52. 37
-PACKET BROADCAST NETWORKS - A PERFORMANCE ANALYSIS OF THE R-ALOHA PROTOCOL
-IEEE TRANS ON COMPUTERS V 29 N 7 P 596

53. 36
-AN INTERNATIONAL EXPERIMENT IN HIGH-SPEED COMPUTER NETWORKING VIA SATELLITE
-COMSAT TECHNICAL REVIEW V.10 N 1 P 27

54. 81-6
-AN ADAPTIVE BUFFER CONTROL SCHEME FOR SLOTTED ALOHA SATELLITE CHANNELS
-COMPUTERS AND ELECTRICAL ENGINEERING, V 7 N 4 P 287

55. 81-6
-DISTRIBUTED SCHEDULING CONFLICT-FREE MULTIPLE ACCESS FOR LOCAL AREA COMMUNICATION NETWORKS
-IEEE TRANS COMMUN V 28 N 12 P 1968

56. 81-6
-CHOICE OF REGULATOR FOR MUTUAL NETWORK SYNCHRONIZATION
-IEEE TRANS COMMUN V 28 N 12 P 1989

57. 81-6
-A SIMPLE MODEL FOR COMPUTATION OF PACKET RADIO NETWORK COMMUNICATION PERFORMANCE
-IEEE TRANS COMMUN V 28 N 12 P 2020

58. 81-6
-THE CONTRACT NET PROTOCOL - HIGH LEVEL COMMUNICATION AND CONTRL IN A DISTRIBUTED PROBLEM SOLVER
-IEEE TRANS COMPUTERS V 29 N 12 P 1104 (A SPECIAL ISSUE ON DISTRIBUTED PROCESSING SYSTEMS

59. 81-6 MICROS, A DISTRIBUTED OPERATING SYSTEM FOR MICRONET, A RECONFIGURABLE NETWORK CONPUTER
-IEEE TRANS COMPUTERS V 29 N 12 P 1133 (A SPECIAL ISSUE ON DISTRIBUTED PROCESSING SYSTEMS
THE SYSTEM IS WRITTEN IN CONCURENT PASCAL FOR A NETWORK OF LSI-11 PROCESSORS

60. 81-6
-ON THE STATUS OF PACKET SWITCHING TECHNOLOGY 1. REVIEW AND 2. EXISTING NETWORKS, ARCHETECTURE OF SWITCHING COMPUTERS
-WISSENSCHAFTLICHE BERICHTE AEG-TELEFUNKEN V 53 N 3 P 111, 126

DESCRIPTION OF MODEM OF VE3DVV

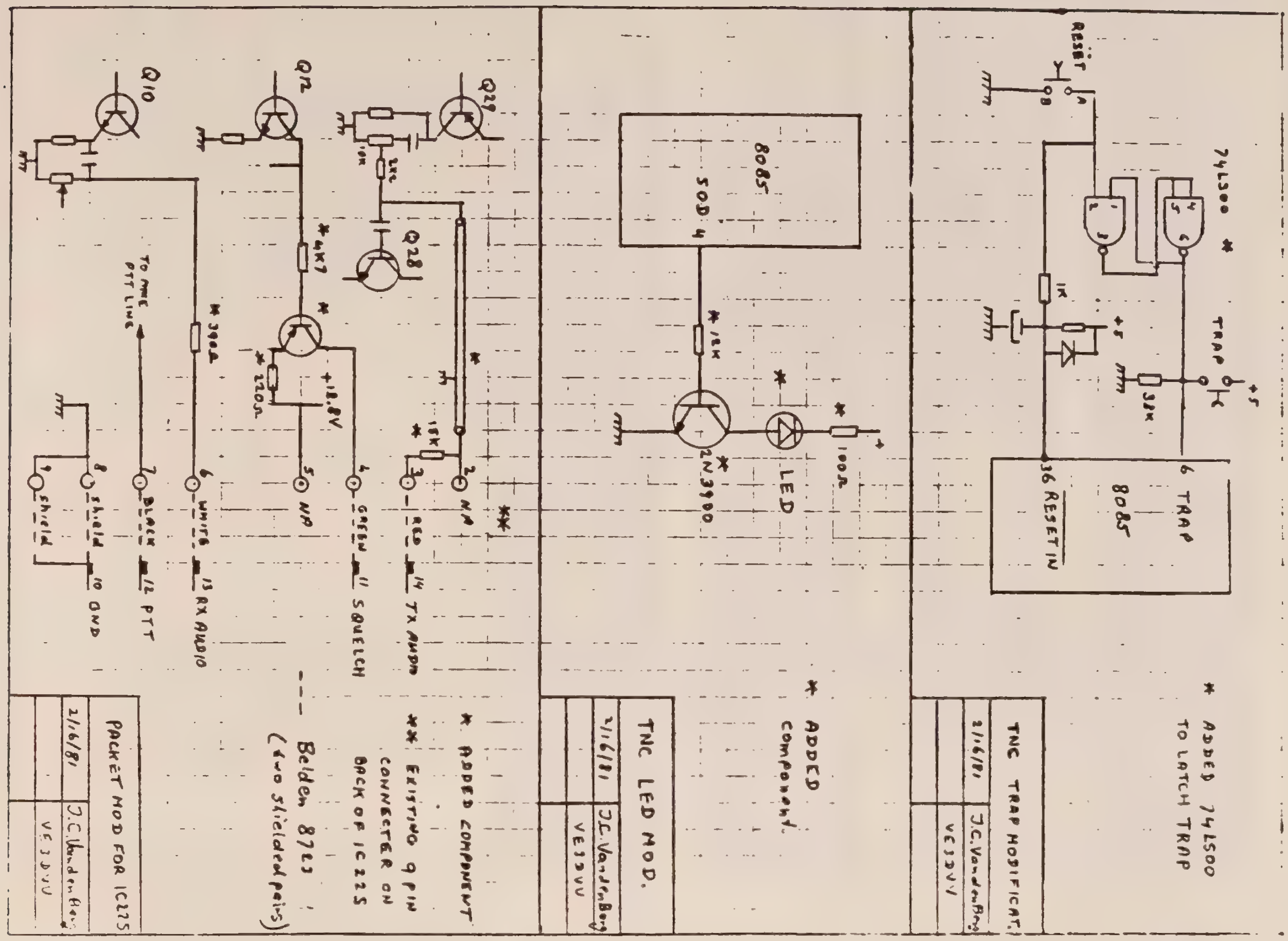
The hart of the modem are the XR2206 sinewave generator, and the XR2211 phaselock demodulator chips.
The BELL 202 compatible tones used are 1200 and 2200 Herz.
Standart RS232 interface levels are used for inerfacing with the T.N.C. . The leds for monitoring the interface signals are quite helpfull for checking line status.
Another feature is the analog switch MC14016 which turns off the modem output when you want to talk over the radio (PTT pressed)
The microphone is here continuesly connected and does not interfere with packet transmissions. Also if the T.N.C. wants to transmit a packet this packet will wait untill you let the mike button go (CTS being delayed).
The dip switch allowes setting the modem for half duplex (normal mode) or full duplex and allowes easy testing. Also S7 allowes setting of an extra long CTS delay for some of our fellow slowpokes with relay swiching.

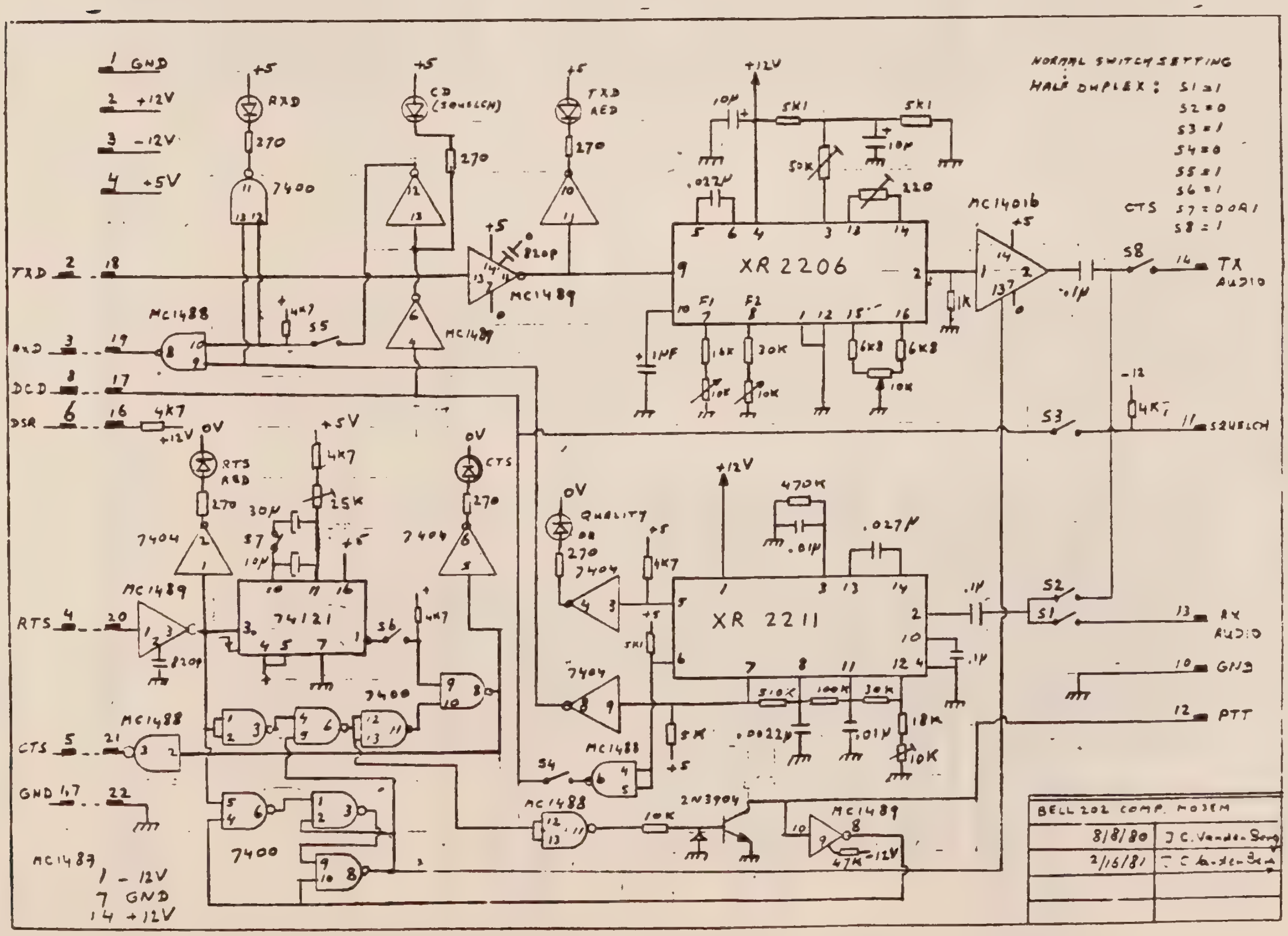
CONSTRUCTION AND ADJUSTMENT :

The modem is build on a small pluggable card with a 22 pin edge connector from my local radio shack and attached to the same power supply as the T.N.C. .
The modem connects to the radio with a shealded cable that, in my case plugs into the exsisting socket of my ICOM 22S which is sligly modified to accommodate the modem.

The adjustments are very simple. First set up the XR2206 tones using a frequency counter, adjusting f1 for 1200 and f2 for 2200 Herz. Then loop the signal back into the demodulator (S9=0, S2=1,S1=0). Now adjust the loop frequency so that with the T.N.C. sending flags (normal initialized condition) till the RXD looks identical to TXD.
The assymetry is not critical and could be substituted with a permanent resistor 150 ohm.
The 50 K pot. should be adjusted for proper devation, here approxymetly 3 Kc. This can easily be checked with a scope at the a frend's receiver.
The setting of the request to send delay (74121) should be done experimentally. For a rig with solid state switching a value around 100 msec is probably ok, for relay switching close S7 and try around 400 msec.

good luck, see you on PACKET RADIO.
(JOHN C. VANDEN BERG)


PACKET MOD FOR IC22S
2/16/81
J.C. VandenBerg
VE3DVU
* ADDED COMPONENT
** EXISTING 9 PIN CONNECTOR ON BACK OF IC22S
--- Belden 8723 (4-2 shielded pairs)
TNC LED MOD.
* ADDED COMPONENT.
TNC TRAP MODIFICATION
* ADDED 74LS00 TO LATCH TRAP
8085
74LS00
RESET IN
TRAP
LED
2N3904


1 GND
2 +12V
3 -12V
4 +5V
NORMAL SWITCH SETTING
HALF DUPLEX: S1=1
S2=0
S3=1
S4=0
S5=1
S6=1
CTS S7=0 OR 1
S8=1
XR 2206
XR 2211
74121
MC14016
TX AUDIO
SQUELCH
RX AUDIO
GND
PTT
MC1489
1 - 12V
7 GND
14 +12V
BELL 202 COMP. MODEM
8/8/80 J.C. VandenBerg
2/16/81 J.C. VandenBerg

THE PROGRAM " TNCDUMP " IS WRITTEN FOR CP/M , BUT COULD BE
MODIFIED FOR OTHER OPERATING SYSTEMS.

The run procedure is as follows:

1. hit trap switch on the T.N.C.,the LED will flash. The T.N.C. is now waiting for a command to dump the data.
2. the host program TNCDUMP should be run next. The program prompts for a comment to add to the dump and time stamp if the clock here is running . If you do not have a clock , the time will be skipped.
3. the program now requests the trap data , and 8K bytes containing all memory (EROM + RAM) and registers is tranferred to the host memory.
4. the program promts for the data to be send to the console , list device or both.
5. the next promt is what part of the data you want to use.
6. the register contents are shown first ,then the memory data in both hex and ascii format.
7. a return to CP/M option returns you to the operating system. The T.N.C. should be reset to get out of the trap routine.

(EDITOR'S NOTE: 'TNCDUMP' WILL APPEAR IN THE NEXT BULLETIN)

##

HERE FOLLOWS AN EXAMPLE OF MY TRAP DUMP.
only RAM 1 is dumped here.

##

```
TTTTTTT  N    N    CCCC        DDDDD    U    U    M     M     PPPPP
   T     NN   N   C            D    D   U    U    MM   MM     P    P
   T     N N  N   C            D    D   U    U    M MM  M     PPPPP
   T     N  N N   C            D    D   U    U    M     M     P
   T     N    N    CCCC        DDDDD     UUUUU    M     M     P
```

TERMINAL CONTROLER TRAP DUMP
DATE : 02/12/81   TIME 22 12 56
NOTE : VE3DSP FILLED LIP BUFFER AND KEEPS RETRANSMITTING THE NEXT DATA WHICH IS NOT BEING ACCEPTED.

```
REGISTERS :
        PC = 18CC        SP = 14EC
        A  = 00          FL = 44          MK = 08
        BC = 001F        DE = 11DE        HL = 11DE
```

```
1000: 60B00000 3F123F12 46121F10 40102010  0040DE11 DE11DE11 E111DD11 06000051  1000: '...?.?.F. .@. ..@..............Q
1020: 04C1FA00 616104C8 00004951 4706E100  00800C17 56453344 56565645 33445350  1020: ....a.....IQG.......VE3DVVVE3DSP
1040: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  1040: ................................
1060: 00000000 00000000 00000000 00000000  00000000 000000C0 00000000 00003301  1060: ..............................3.
1080: 410D0A27 00610439 30204124 3D224C4F  4E474954 55444522 203A2047 4F535542  1080: A..'.a.90 A$="LONGITUDE" : GOSUB
10A0: 20313030 30203A20 44343D41 0D0A2D00  61163130 30202720 434F4D50 55544520  10A0:  1000 : D4=A..-.a.100 ' COMPUTE
10C0: 414E474C 45204320 41532044 49464620  4F46204C 4F4E4749 54554445 530D0A13  10C0: ANGLE C AS DIFF OF LONGITUDES...
10E0: 00611831 31302043 323D2844 342D4432  292F322E 0D0A1700 610A3131 35204946  10E0: .a.110 C2=(D4-D2)/2.....a.115 IF

1100: 43323D30 2E302054 48454E20 3430300D  0A370061 0C313230 20272043 4F4D5055  1100: C2=0.0 THEN 400..7.a.120 ' COMPU
1120: 54452048 414C4620 414E474C 45532046  4F522044 49464620 26205355 4D204F46  1120: TE HALF ANGLES FOR DIFF & SUM OF
1140: 204C4154 49545544 45530D0A 2300610E  31333020 4C443D28 44332D44 31292F32  1140:  LATITUDES..#.a.130 LD=(D3-D1)/2
1160: 2E203A20 4C533D28 44332B44 31292F32  2E0D0A00 00000148 00610011 33352050  1160: . : LS=(D3+D1)/2.......H.a.135 P
1180: 52494E54 22494E46 4F2D2D20 432F323D  223B4332 2A523244 2C224C41 54205355  1180: RINT"INFO-- C/2=";C2*R2D,"LAT SU
11A0: 4D2F323D 223B4C53 2A523244 2C224C41  54204449 46462F32 3D223B4C 442A5232  11A0: M/2=";LS*R2D,"LAT DIFF/2=";LD*R2
11C0: 440D0A17 00610231 34302041 3D434F53  28433229 2F53494E 28433229 0D0A313D  11C0: D....a.140 A=COS(C2)/SIN(C2)..1=
11E0: 410D3127 00610037 30204124 3D224C4F  4E474954 55444522 203A2047 4F535542  11E0: A.1'.a.70 A$="LONGITUDE" : GOSUB

1200: 20313030 30203A20 44323D41 0D0A3000  61023830 2041243D 224C4154 49545544  1200:  1000 : D2=A..0.a.80 A$="LATITUD
1220: 45202220 3A204224 3D224222 203A2047  4F535542 20313030 30203A20 44333D00  1220: E " : B$="B" : GOSUB 1000 : D3=.
1240: 00564533 44535000 00000000 00000000  00000000 00000000 00000000 00000000  1240: .VE3DSP.........................
1260: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  1260: ................................
1280: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  1280: ................................
12A0: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  12A0: ................................
12C0: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  12C0: ................................
12E0: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  12E0: ................................

1300: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  1300: ................................
1320: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  1320: ................................
1340: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  1340: ................................
1360: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  1360: ................................
1380: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  1380: ................................
13A0: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  13A0: ................................
13C0: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  13C0: ................................
13E0: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 000000ED  13E0: ................................
```

```
                    ;**********************************************
                    ;     TIP TNC TRAP DUMP MODULE
                    ;**********************************************
                    ;
                    ;     THIS PROGRAM IS ENTERED ON ACTIVATING
                    ;     THE TRAP SWITCH OF THE T.N.C.
                    ;
                    ;     THE PROGRAM SAVES THE REGISTERS AND
                    ;     INITIALIZES THE WD8250 HOST INTERFACE
                    ;
                    ;     THEN THE T.N.C. WILL SIT WITH THE
                    ;     INDICATOR LED FLASHING WAITING FOR
                    ;     THE HOST TO SEND THE TRANSFER REQUEST
                    ;
                    ;     WHEN THE TRANSFER REQUEST IS RECEIVED
                    ;     ALL 8K BYTES OF MEMORY ARE SEND TO THE
                    ;     HOST AND THE TRANSFER REQUEST LOOP IS
                    ;     ENTERED AGAIN.
                    ;
                    ;     TO TERMINATE HIT THE T.N.C. RESET
                    ;
                    ;     01/01/81      J.C.VANDEN BERG
                    ;     01/10/81      ADDED FLASHING LED
                    ;
                    ;=============================================
                    ;
                    ;
                    .PABS
                    ;
                    .DEFINE RIM    =     [.BYTE  20H]
                    .DEFINE SIM    =     [.BYTE  30H]
                    ;
                    ;
0000                UART    =     0H      ;8250 BASE ADDRESS
000B                UARTM   =     0BH     ;8 DATA BITS, 1STOP BIT, ODD PARI
0008                DIVISOR =     8       ;FOR 19.200 BAUD
0000                REG0    =     UART
0001                REG1    =     UART+1
0002                REG2    =     UART+2
0003                REG3    =     UART+3
0004                REG4    =     UART+4
0005                REG5    =     UART+5
0006                REG6    =     UART+6
                    ;
                    ;
0C00                .LOC    0C00H   ;START OF EPROM
                    ;
                    ; FILL FIRST PART OF EPROM WITH 0FFH
                    ;
0C00                LOC00H:
0C00  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
```



```
0C08  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C10  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C18  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C20  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C28  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C30  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C38  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C40  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C48  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C50  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C58  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C60  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C68  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C70  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C78  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C80  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C88  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C90  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0C98  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CA0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CA8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CB0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CB8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CC0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CC8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CD0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CD8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CE0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CE8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CF0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0CF8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
                    ;
0D00                LOD00H:
0D00  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D08  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D10  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D18  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D20  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D28  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D30  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D38  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D40  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D48  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D50  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D58  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D60  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D68  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D70  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D78  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D80  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D88  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
```



```
0D90  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0D98  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DA0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DA8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DB0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DB8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DC0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DC8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DD0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DD8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DE0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DE8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DF0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0DF8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
                    ;
0E00                LOE00H:
0E00  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E08  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E10  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E18  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E20  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E28  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E30  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E38  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E40  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E48  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E50  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E58  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E60  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E68  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E70  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E78  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E80  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E88  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E90  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0E98  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EA0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EA8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EB0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EB8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EC0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EC8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0ED0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0ED8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EE0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EE8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EF0  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0EF8  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
                    ;
0F00                LOF00H:
0F00  FFFF FFFF           .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
```

reversed by mistake!


```
                ; KEEP LED FLASHING
                ;
0FB4  DB05      WAIT:   IN      REG5    ;WAIT FOR RECEIVE READY
0FB6  E601              ANI     1
0FB8  C2 0FBC           JNZ     INPUT   ;READ THE CHAR FROM HOST

                ; FLASH LED ON
0FBB  3EC0      WAIT1:  MVI     A,0C0H
0FBD  30                SIM             ;FLASH LED ON[.BYTE    30H]
0FBE  2620              MVI     H,20H
0FC0  2E00      ..LP0:  MVI     L,0
0FC2  2D        ..LP1:  DCR     L
0FC3  C2 0FC2           JNZ     ..LP1
0FC6  25                DCR     H
0FC7  C2 0FC0           JNZ     ..LP0

                ; FLASH LED OFF
0FCA  3E40      WAIT2:  MVI     A,040H
0FCC  30                SIM             ;FLASH LED OFF[.BYTE    30H]
0FCD  2620              MVI     H,20H
0FCF  2E00      ..LP0:  MVI     L,0
0FD1  2D        ..LP1:  DCR     L
0FD2  C2 0FD1           JNZ     ..LP1
0FD5  25                DCR     H
0FD6  C2 0FCF           JNZ     ..LP0
0FD9  C3 0FB4           JMP     WAIT

0FDC  DB00      INPUT:  IN      REG0    ;GET INPUT DATA FROM HOST
0FDE  FE2A              CPI     '*'
0FE0  C2 0FB4           JNZ     WAIT
0FE3  C3 0FE6           JMP     DUMP

                ;
                ; DUMP THE MEMORY AND SEND TO HOST
                ;
0FE6  21 0000   DUMP:   LXI     H,0     ;START OF MEMORY
0FE9  DB05      ..WT2:  IN      REG5    ;WAIT FOR TRANSMIT READY
0FEB  E620              ANI     20H
0FED  CA 0FE9           JZ      ..WT2
0FF0  7E                MOV     A,M     ;GET THE DATA
0FF1  D300              OUT     REG0    ;AND SEND IT
0FF3  23                INX     H       ;UPDATE POINTER
0FF4  7C                MOV     A,H
0FF5  FE30              CPI     30H     ;STOP IF 3000H
0FF7  C2 0FE9           JNZ     ..WT2

0FFA  C3 0FB4           JMP     WAIT    ;ALL DONE, FLASH AGN
                ;
                ;
                ;########## TRAP VECTOR ####################
                ;
```



```
0F08  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F10  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F18  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F20  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F28  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F30  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F38  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F40  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F48  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F50  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F58  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F60  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F68  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F70  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F78  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
                ;
0F80            LOF80H:
0F80  FFFF FFFF         .WORD   0FFFFH,0FFFFH,0FFFFH,0FFFFH
0F88  FFFF FFFF         .WORD   0FFFFH,0FFFFH
                ;
0F8C            .LOC    0F8CH
                ;
                ; DISABLE INTERUPTS SAVE CPU REGISTERS
                ;
0F8C            TIPDMP1:
0F8C  22 1FF6           SHLD    TSAVHL  ;SAVE HL
0F8F  21 0000           LXI     H,0
0F92  39                DAD     SP
0F93  22 1FF4           SHLD    TSAVSP  ;SAVE STACK POINTER
0F96  E1                POP     H       ;NOW PC IN HL
0F97  31 2000           LXI     SP,TSAVPC+2
0F9A  E5                PUSH    H       ;SAVE PC
0F9B  D5                PUSH    D       ;SAVE DE
0F9C  C5                PUSH    B       ;SAVE BC
0F9D  F5                PUSH    PSW     ;SAVE PSW
0F9E  20                RIM[.BYTE       20H]
0F9F  32 1FF3           STA     TSAVM   ;SAVE MASK
                ;
                ;  INITIALIZE WD8250 INTERFACE
                ;
0FA2  3E80              MVI     A,80H
0FA4  D303              OUT     REG3    ;SET DIVISOR ACCESS BIT
0FA6  3E08              MVI     A,DIVISOR&0FFH
0FA8  D300              OUT     REG0
0FAA  3E00              MVI     A,DIVISOR>8
0FAC  D301              OUT     REG1
0FAE  3E0B              MVI     A,UARTM ;SET MODE
0FB0  D303              OUT     REG3
0FB2  DB00              IN      REG0    ;CLEAR GARBAGE
                ;
                ; WAIT FOR DUMP COMMAND FROM HOST AND
```



```
0FFD            .LOC    0FFDH
                ;
0FFD  C3 0F8C           JMP     TIPDMP1
                ;
                ;==========================================
                ;       REGISTER SAVE AREA
                ;==========================================
                ;
1FF3            .LOC    1FF3H
                ;
1FF3            TSAVM:          .BLKB   1       ;MASK SAVE
1FF4            TSAVSP:         .BLKW   1       ;SAVED SP
1FF6            TSAVHL:         .BLKW   1       ;SAVED HL
1FF8            TSAVPSW:        .BLKW   1       ;SAVED PSW
1FFA            TSAVBC:         .BLKW   1       ;SAVED BC
1FFC            TSAVDE:         .BLKW   1       ;SAVED DE
1FFE            TSAVPC:         .BLKW   1       ;SAVED PC
                ;
                .END
```

The T.N.C. board here is slightly modified to use the trap function and a LED has been added to indicate status.

The existing trapswitch circuit creates bounce that makes saving the registers impossible. Also there is the land across R7 that has to be cut.

To eliminate the bounce problem I added an 7400LS to latch the trap status. The flipflop is reset by using the reset button. (see diagram 1)
The IC is physically located on the board, below J4.

The LED is driven by a transistor from the SIO line of the 8085. (see diagram 2 , from VE3EC)
The LED and transistor are located here on the board at CR3 and Q1 , the resistors using the holes of R18 and CR2. Some traces had to be cut.

Some uses for which I use the LED are :

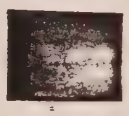
1. it blinks a couple of times on P.O.R. to indicate the LIP has been initialized.
2. it indicates an error if it blinks during a download.
3. it blinks 6 times on a tip reset.(program reset)
4. it blinks 6 times on hardware reset if my TIP is used.
5. it turns on continuesly when a station is connected.
6. it will flash fast continuesly if the trap switch is used and trap data is available.

J.C.VANDENBERG (VE3DVV)

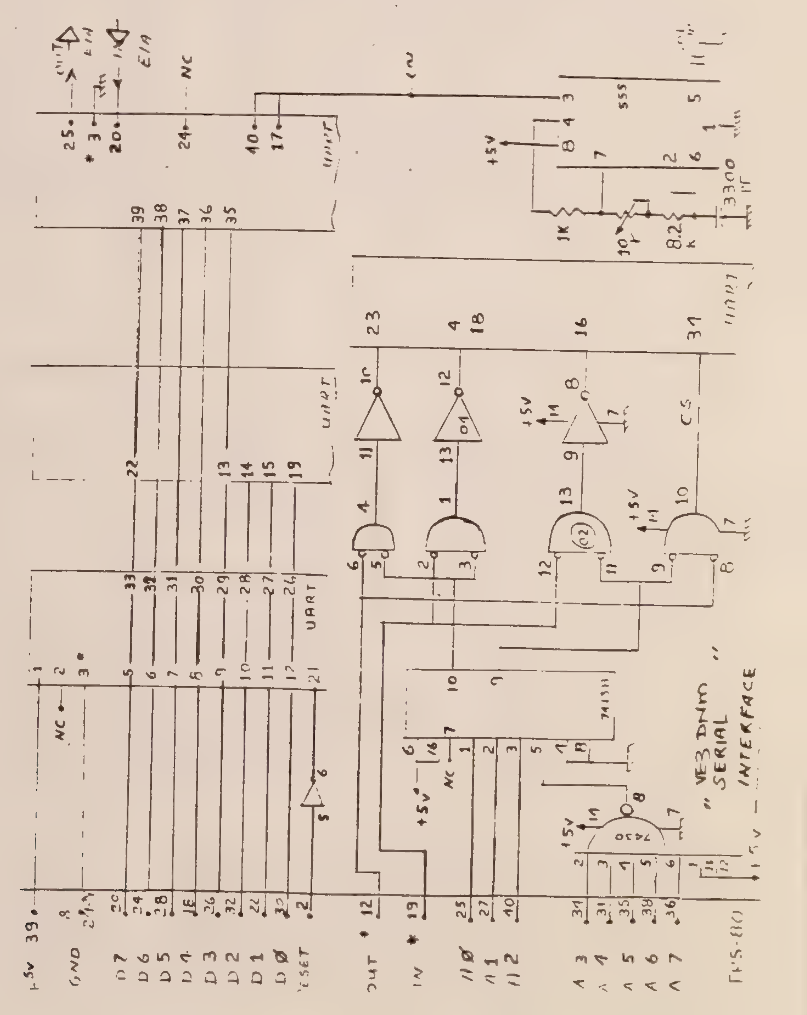
```
;       ROUTINE TO DRIVE UART INTERFACE
;       GENERAL PACKET CONTROL    VERS 3.1
        ORG     07B00H  ;ASSIGN START
DPORT   EQU     0FDH    ;DATA PORT OF UART
SPORT   EQU     0FEH    ;STATUS & CONTROL PORT
BASIC   EQU     0       ;COLD START BASIC TO FIX PTRS
TSCRN   EQU     3C00H   ;TOP OF SCREEN
BLANK   EQU     20H     ;ASCII BLANK CODE
CURSOR  EQU     4020H   ;CURSOR POINTER
ENTER   EQU     0DH     ;ENTER KEY
BREAK   EQU     1       ;BREAK KEY
LF      EQU     0AH     ;DOWN ARROW KEY
VIDEO   EQU     33H     ;BASIC SCREEN HANDLER
LINE2   EQU     3C40H   ;2ND LINE OF SCREEN
KEYB    EQU     2BH     ;BASIC KEYBOARD SCAN
USCORE  EQU     5FH     ;UNDERSCORE CURSOR CHAR.
CLEAR   EQU     1FH     ;CODE TO CLEAR SCREEN
FFEED   EQU     0CH     ;FORM FEED CODE
ESC     EQU     1BH     ;ESCAPE CODE
;
;       GENERAL PACKET CONTROL PROGRAM
;
PACK    CALL    INIT    ;SET UP UART
        CALL    KLEAR   ;CLEAR THE SCREEN
        LD      HL,MESS
        LD      DE,TSCRN
        LD      BC,MESS1-MESS
        LDIR
        LD      HL,LINE2        ;SET CURSOR TO LINE 2
        LD      A,USCORE ;PUT CURSOR ON SCREEN
        LD      (HL),A
        LD      (CURSOR),HL
;
;       SETUP IS FINISHED. GO INTO GENERAL OPERATION
;
CHCHK   CALL    RECV    ;HAS ANYTHING BEEN RECEIVED
        JR      NZ,SHRCV ;IF SO, SHOW IT
        CALL    KEYB    ;ANYTHING TO SEND?
        OR      A
        JR      Z,CHCHK
        CP      BREAK   ;DOES USER WANT OUT
        JP      Z,BASIC
        CP      CLEAR   ;IS IT A CLEAR SCREEN REQUEST
        JR      NZ,TESC
        CALL    KLEAR   ;CLEAR THE SCREEN
        JR      CHCHK
TESC    CP      ESC     ;TEST FOR ESCAPE CODE
        JR      NZ,SCHAR
        LD      A,FFEED ;SUB FORM FEED
SCHAR   CALL    KSEND   ;SEND IT
        CP      ENTER   ;IS IT THE ENTER KEY
        JR      NZ,CHCHK
SREQ    LD      A,LF    ;ISSUE SEND REQUEST
        CALL    KSEND
        JR      CHCHK
SHRCV   PUSH    AF      ;SAVE FLAGS
        CALL    C,CERROR
        POP     AF
        PUSH    AF
        CALL    M,FERROR
        POP     AF
        CP      LF      ;CHECK FOR A FORM FEED
        JR      Z,CHCHK ;IF SO, IGNORE IT.
        CP      FFEED   ;IS IT A FORM FEED?
        JR      NZ,SHOW
        CALL    KLEAR   ;CLEAR THE SCREEN
        JR      CHCHK
SHOW    CALL    VIDEO
        LD      HL,(CURSOR)
        LD      A,40H   ;CHECK FOR SCREEN OVERFLOW
        CP      H
        JR      NZ,CHCHK
        LD      HL,TSCRN
        LD      (CURSOR),HL
        JR      CHCHK
```

```
;
CERROR  LD      HL,MESS1
        LD      DE,TSCRN
        LD      BC,MESS2-MESS1
        LDIR
        RET
FERROR  LD      HL,MESS2
        LD      DE,TSCRN+20H
        LD      BC,COUNT-MESS2
        LDIR
        RET
;
INIT    LD      A,6BH   ;8 DATA & 1 STOP BIT, NO PARIT
        OUT     (SPORT),A
        IN      A,(DPORT) ;CLEAR STATUS REG.
NFRAME  LD      A,240   ;INITIALIZE BYTE COUNTER
        LD      (COUNT),A
        RET
KSEND   CALL    SEND    ;SEND BYTE
        PUSH    AF      ;SAVE IT
        CP      LF      ;WAS IT A SEND REQUEST?
        JR      NZ,KCHAR
ASEND   LD      A,LF    ;AUTOMATIC SEND REQUEST
        CALL    SEND
        CALL    NFRAME  ;RESET BYTE COUNT
        JR      RSTOR
KCHAR   LD      A,(COUNT) ;GET BYTE COUNTER
        DEC     A       ;COUNT IT
        LD      (COUNT),A ;PUT IT BACK
        JR      Z,ASEND ;IF COUNT IS ZERO SEND BUFFER
RSTOR   POP     AF      ;GET BYTE BACK
        RET
SEND    PUSH    AF      ;SAVE BYTE
WAITX   IN      A,(SPORT) ;CHECK IF XMIT BUF IS FULL
        BIT     7,A     ;CHECK 'TBE'
        JR      Z,WAITX ;WAIT FOR IT TO EMPTY
        POP     AF      ;GET BYTE BACK
        OUT     (DPORT),A ;SEND IT
        RET
RECV    IN      A,(SPORT) ;CHECK IF DATA AVAILABLE
        BIT     0,A     ;CHECK BIT
        RET     Z       ;RETURN IF NO DATA AVAIL.
        LD      B,A     ;SAVE CONTENTS
        XOR     A       ;ZERO REG A
        BIT     1,B     ;CHECK FOR OVER RUN
        JR      Z,CHKFRM ;IF OVER RUN SET CARRY
        SET     0,A     ;FOR CARRY FLAG
CHKFRM  BIT     2,B     ;CHECK FRAMING
        JR      Z,SETDAT ;IF FRAMING ERROR
        SET     7,A     ;FOR SIGN FLAG
SETDAT  LD      C,A
        IN      A,(DPORT) ;GET DATA BYTE
        LD      B,A     ;MOVE IT TO SAVE POS.
        PUSH    BC      ;MOVE FLAG STATUS
        POP     AF      ;TO FLAG REG
        RET
;
KLEAR   LD      HL,TSCRN ;ROUTINE TO CLEAR SCREEN
        LD      (CURSOR),HL ;SET POINTER
        LD      A,BLANK
        LD      (HL),A
        LD      DE,TSCRN+1
        LD      BC,400H
        LDIR
        RET
MESS    DEFM    'PACK MODE READY '
MESS1   DEFM    '*** OVER RUN ERROR *** '
MESS2   DEFM    '*** FRAMING ERROR *** '
COUNT   DEFB    0
        END     PACK
```

```
;  ROUTINE TO SEND FROM BASIC TO THE TTY PACKET
;  TIP.  WARNING.  CHARACTERS WILL BE DROPPED UNLESS
;  YOU MAKE SURE TO NEVER FULLY FILL THE SEND BUFFER.
;  EXAMPLE:  ONLY  'LLIST' A FEW LINES AT A TIME
;                  'LPRINT' WITH LOTS OF DEAD TIME.
;              BY  GLENN        VE3DSP   VER 2.1
        ORG     7F80H   ;TTY ENTRY ADDRESS
DPORT   EQU     0FDH    ;UART DATA PORT
SPORT   EQU     0FEH    ;UART CONTROL & STATUS
PADDR   EQU     4026H   ;PRINTER ADDRESS FOR BASIC
LF      EQU     0AH     ;LINE FEED CODE
BASIC   EQU     1A19H   ;WARM START
;
;       SET UART FOR 1200 BAUD
;
LPACK   LD      A,68H   ;8 BIT DATA 1 STOP NP
        OUT     (SPORT),A
        IN      A,(DPORT) ;CLEAR RX BUFFER
        LD      HL,SEND ;SET BASIC POINTER
        LD      (PADDR),HL
        JP      BASIC   ;GIVE CONTROL TO BASIC
;
;  NOTE:   DATA FROM THE TIP IS IGNORED
;
SEND    LD      A,C     ;GET BYTE
        CP      0DH     ; CR?
        JR      Z,CARET
        CP      20H     ;RET IF A CONTROL CODE
        RET     M
        CALL    BYOUT   ;SEND THE BYTE
        PUSH    AF      ;SAVE REGS
        PUSH    BC

 WAIT FOR EITHER A CHAR TO ECHO -OR- A TIME OUT

        LD      BC,8000H ;SET MAX DELAY COUNT
LOOP    IN      A,(SPORT) ;SEE IF CHAR ECHOED
        BIT     7,A
        JR      NZ,CONT
        DJNZ    LOOP
        DEC     C
        JR      NZ,LOOP
CONT    POP     BC
        POP     AF
        RET

 SUBROUTINES OF THE MAIN SUBROUTINE

NLINE   LD      A,0DH
CARET   CALL    BYOUT   ;PRINT CR
        LD      A,LF    ;ADD A LF
BYOUT   PUSH    AF      ;SAVE BYTE
WAIT    IN      A,(SPORT) ;GET TX STATUS
        BIT     7,A     ;IS TX BUFFER EMPTY?
        JR      Z,WAIT
        POP     AF
        OUT     (DPORT),A ; PRINT THE BYTE
        RET
        END     LPACK
```

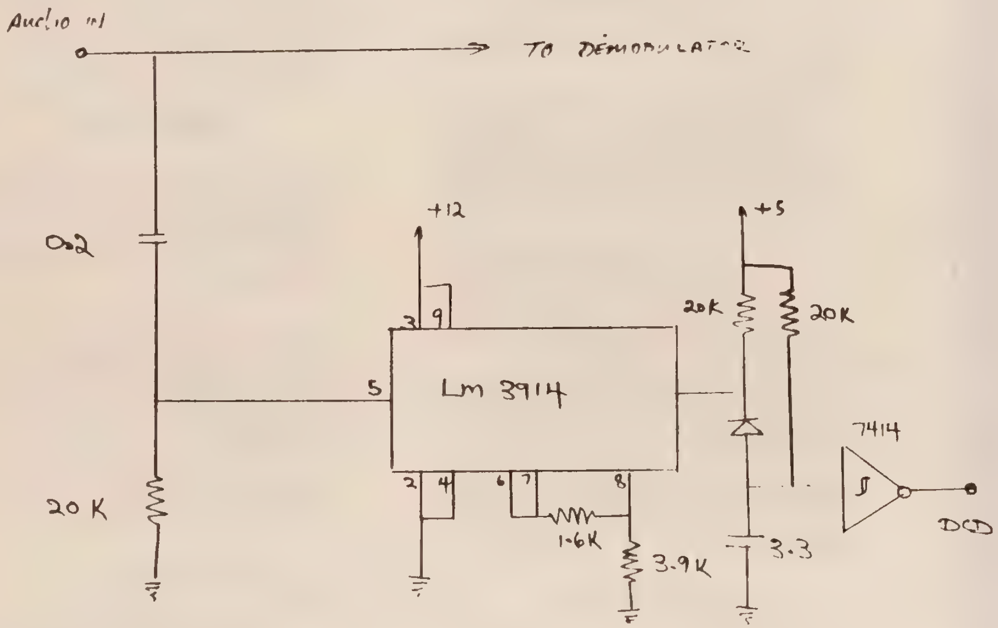


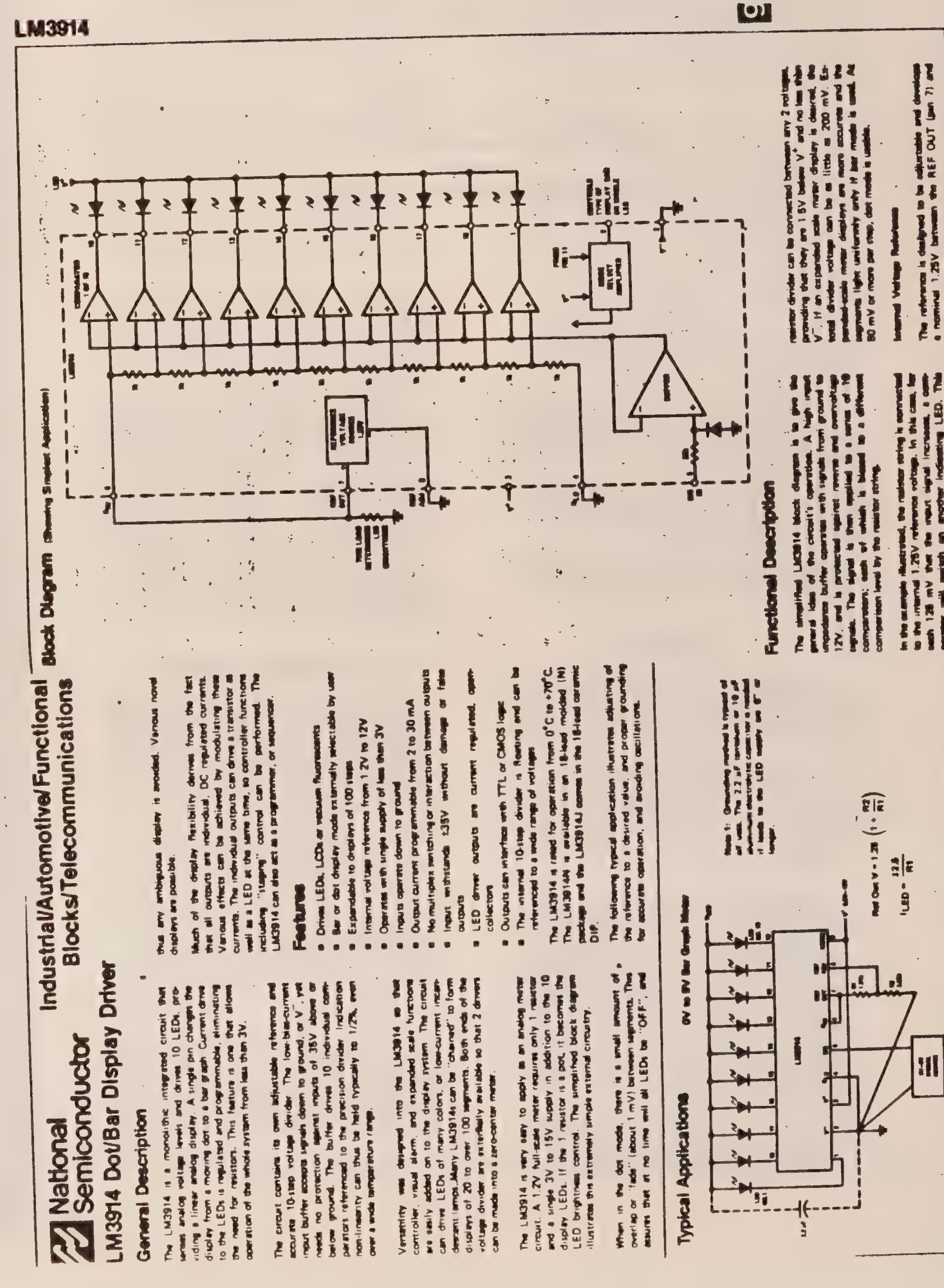
```
;  ROUTINE TO SEND AT 110 BAUD FROM TRS 80 AT 110 BAUD
        ORG     7F80H   ;TTY ENTRY ADDRESS
DPORT   EQU     0FDH    ;UART DATA PORT
SPORT   EQU     0FEH    ;UART CONTROL & STATUS
PADDR.  EQU     4026H   ;ADDRESS PTR FOR BASIC
BASIC   EQU     1A19H   ;WARM START
;
TTY     LD      A,74H   ;SET 8 BIT DATA 2 STOP NP
        OUT     (SPORT),A
        IN      A,(DPORT) ;CLEAR RX BUFFER
        LD      HL,SEND ;SET BASIC POINTER
        LD      (PADDR),HL
        JP      BASIC   ;GIVE CONTROL TO BASIC
SEND    LD      A,C     ;GET BYTE
        CP      0DH     ; CR?
        JR      Z,CARET
        CP      20H     ;RET IF A CONTROL CODE
        RET     M
        PUSH    AF
        LD      A,(COLNO) ;FIND OUT WHAT COL. #
        OR      A
        CALL    Z,NLINE ;IF FULL THEN NEW LINE
        POP     AF
        CALL    BYOUT   ;PRINT THE BYTE
        PUSH    HL
        LD      HL,COLNO ;COUNT THE BYTE
        DEC     (HL)
        POP     HL
        LD      A,C
        RET
NLINE   LD      A,0DH   ;ISSUE CR
CARET   CALL    BYOUT   ;PRINT CR
        LD      A,70    ;SET NEW LINE COUNT
        LD      (COLNO),A
        LD      A,0AH   ;ADD A LF
BYOUT   PUSH    AF      ;SAVE BYTE
WAIT    IN      A,(SPORT) ;GET TX STATUS
        BIT     7,A     ;IS TX BUFFER EMPTY?
        JR      Z,WAIT
        POP     AF      ;IT EMPTY SO
        OUT     (DPORT),A ; PRINT THE BYTE
        RET
COLNO   DEFB    70
        END     TTY
```



The following is a circuit used by Robert Sleath, VE3EFD, to generate 'DCD' instead of digging into the guts of his handheld tranceiver. It uses a National LM3914 Dot/Bar display driver (usually available from Radio Shack for less than $5. The audio level is converted by the internal comparators into a series of LED driving levels.

The output from the lowest indicator is fed into a RC circuit to eliminate short transients and then thru a pair of schmidt triggers to provide switching hysteresis.





LM3914

National Semiconductor

Industrial/Automotive/Functional Blocks/Telecommunications

LM3914 Dot/Bar Display Driver

General Description

The LM3914 is a monolithic integrated circuit that senses analog voltage levels and drives 10 LEDs, providing a linear analog display. A single pin changes the display from a moving dot to a bar graph. Current drive to the LEDs is regulated and programmable, eliminating the need for resistors. This feature is one that allows operation of the whole system from less than 3V.

The circuit contains its own adjustable reference and accurate 10-step voltage divider. The low-bias-current input buffer accepts signals down to ground, or V^-, yet needs no protection against inputs of 35V above or below ground. The buffer drives 10 individual comparators referenced to the precision divider. Indication non-linearity can thus be held typically to 1/2%, even over a wide temperature range.

Versatility was designed into the LM3914 so that controller, visual alarm, and expanded scale functions are easily added on to the display system. The circuit can drive LEDs of many colors, or low-current incandescent lamps. Many LM3914s can be "chained" to form displays of 20 to over 100 segments. Both ends of the voltage divider are externally available so that 2 drivers can be made into a zero-center meter.

The LM3914 is very easy to apply as an analog meter circuit. A 1.2V full-scale meter requires only 1 resistor and a single 3V to 15V supply in addition to the 10 display LEDs. If the 1 resistor is a pot, it becomes the LED brightness control. The simplified block diagram illustrates this extremely simple external circuitry.

When in the dot mode, there is a small amount of overlap or "fade" (about 1 mV) between segments. This assures that at no time will all LEDs be "OFF", and thus any ambiguous display is avoided. Various novel displays are possible.

Much of the display flexibility derives from the fact that all outputs are individual, DC regulated currents. Various effects can be achieved by modulating these currents. The individual outputs can drive a transistor as well as a LED at the same time, so controller functions including "staging" control can be performed. The LM3914 can also act as a programmer, or sequencer.

Features

- Drives LEDs, LCDs or vacuum fluorescents
- Bar or dot display mode externally selectable by user
- Expandable to displays of 100 steps
- Internal voltage reference from 1.2V to 12V
- Operates with single supply of less than 3V
- Inputs operate down to ground
- Output current programmable from 2 to 30 mA
- No multiplex switching or interaction between outputs
- Input withstands ±35V without damage or false outputs
- LED driver outputs are current regulated, open-collectors
- Outputs can interface with TTL or CMOS logic
- The internal 10-step divider is floating and can be referenced to a wide range of voltages

The LM3914 is rated for operation from 0°C to +70°C. The LM3914N is available in an 18-lead molded (N) package and the LM3914J comes in the 18-lead ceramic DIP.

The following typical application illustrates adjusting of the reference to a desired value, and proper grounding for accurate operation, and avoiding oscillations.

Typical Applications

0V to 5V Bar Graph Meter

Note 1: Grounding method is typical of all uses. The 2.2 μF tantalum or 10 μF aluminum electrolytic capacitor is needed if leads to the LED supply are 6" or longer.

Ref Out $V = 1.25\left(1 + \frac{R2}{R1}\right)$

$I_{LED} = \frac{12.5}{R1}$

Block Diagram (Showing Simplest Application)

Functional Description

The simplified LM3914 block diagram is to give the general idea of the circuit's operation. A high input impedance buffer operates with signals from ground to 12V, and is protected against reverse and overvoltage signals. The signal is then applied to a series of 10 comparators; each of which is biased to a different comparison level by the resistor string.

In the example illustrated, the resistor string is connected to the internal 1.25V reference voltage. In this case, for each 125 mV that the input signal increases, a comparator will switch on another indicating LED. This resistor divider can be connected between any 2 voltages, providing that they are 1.5V below V^+ and no lower than V^-. If an expanded scale meter display is desired, the total divider voltage can be as little as 200 mV. Expanded-scale meter displays are more accurate and the segments light uniformly only if bar mode is used. At 50 mV or more per step, dot mode is usable.

Internal Voltage Reference

The reference is designed to be adjustable and develops a nominal 1.25V between the REF OUT (pin 7) and

FROM: 1980 INTERNATIONAL ZURICH SEMINAR ON DIGITAL COMMUNICATIONS
DIGITAL TRANSMISSION IN WIRELESS SYSTEMS
MAR 4-6 1980, ZURICH

PACKET SPEECH FOR LAND MOBILE CHANNELS

J. Schwarz da Silva, J. B. de Mercado
Ministry of Communications, Ottawa, Ontario, Canada
S. Mahmoud
Carleton University, Ottawa, Ontario, Canada

ABSTRACT

This paper describes the results of a study to evaluate the feasibility of applying packet speech concepts to land mobile systems.

One particular random access technique (NPCSMA) is investigated and it is shown that for typical values of the system parameters packet speech compares favourably to conventional trunked land mobile system.

An expression is derived for the maximum number of active users that can be supported by a single 30 KHz channel, as a function of the voice digitization rate and the lost packet level.

It is further shown that the number of base-to-mobile channels does not need to be equal to the number of mobile-to-base channels.

INTRODUCTION

In the past few years considerable interest has been demonstrated in digital voice techniques /1, 2/, packet radio random access schemes /3, 4/ and cellular structures for land mobile communications /5, 6/.

A number of experiments have been carried out over the ARPANET /7/, to demonstrate the feasibility of transmitting packetized voice. Also the integration of packetized data and packetized voice has been advocated by some researchers /8/ who have attempted to quantify the performance measures of such integrated networks and provide some design guidelines.

Closely related to the idea of packetized voice is the concept of speech interpolation which permits a number of voice sources to share a number of channels through voice-activated switching by taking advantage of the gaps or pauses that naturally occur during a conversation. One of the earliest /9/ speech interpolation systems known as TASI (Time Assignment Speech Interpolation) was a pure analog system whose performance parameter was the fraction of speech lost due to freeze-outs. A freeze-out occurs when a given talkspurt finds no channel idle. Recently a digital version of speech interpolation known as DSI has been advocated in particular for satellite circuits /10/.

On the other hand a considerable body of knowledge /11, 12, 13/ exists for the so-called packet radio systems whose potential have been exploited uniquely for data transmission purposes.

Packet radio concepts have also been applied to the control channels of the Chicago /14/ and Tokyo /15/ cellular systems. It has also been suggested by Norway /16/ that packet radio concepts could be efficiently applied to the signalling channels of the maritime mobile service.

However, no attempt has so far been made to apply the concepts of digital speech interpolation and packetized voice to increase the traffic carrying capacity of land mobile systems.

In this paper we show that for typical values of the system parameters, a land mobile system that employs packetized voice concepts, compares favourably to the conventional trunked land mobile system. We derive an expression for the maximum number of active users that can be supported by a single 30 KHz uplink channel (mobiles to base) and show that the number of downlink (base to mobiles) channels does not need to be equal to the number of uplink channels.

OPERATIONAL CONCEPT

Consider a conventional land mobile system where a large number of speech sources want to exchange voice communication through a finite number of radio channels available at a given base station. If we assume an Erlang B traffic model, the grade of service or blocking probability is given by:

$$P_B = \frac{\rho^M/M!}{\sum_{n=0}^{M} \rho^n/n!} \tag{1}$$

where "ρ" is the traffic offered to the group of "M" channels by the population of users. The total traffic carried by these channels is therefore:

$$\rho_c = \rho\,(1-P_B) \tag{2}$$

The number of users that can be supported is easily obtained by dividing "ρ_c" by the traffic load per voice source. If as an example we consider 100 channels, a blocking probability of 0.02 and a load per voice source of 0.02 erlangs, we conclude that approximately 49 users per channel could be supported.

The same channels could be shared, perhaps more efficiently, by (1) taking advantage of the

statistical characteristics of voice (2) digitizing the talkspurts and encoding them into packets of fixed size and (3) transmitting these packets at high speed. As shown in the model of Figure (1) a very large number of speech sources could share the channel if it were possible to perfectly utilize the pauses in the conversations. It is also clear from the same figure that in such an ideal case the number of speech sources that could share the channel would be a function of the talkspurt length distribution, the pause length distribution, the encoding rate (R), the packet size (B), the overhead (b) and the transmission speed (C).

In practice it is however, virtually impossible to perfectly schedule the activity of the channel without resorting to a centralized control mechanism. A possible alternative is to use a form of distributed control, which simply means that a packet from a given user could be prevented from being transmitted if the channel happened to be busy. If however at some point in time the channel is idle, it is quite possible that nobody else is in the process of transmitting a voice packet.

Fortunately enough such access protocols have already been extensively studied /17/, in the context of packetized data transmission. One of these protocols known as the Non-Persistent Carrier Sense Multiple Access (NPCSMA) scheme operates as follows:

- If a terminal has a packet ready for transmission it senses the channel and if the channel is sensed idle, the packet is transmitted. It can however, collide with some other packet during a time window, which is related to the propagation delay between terminals.
- If the channel is sensed busy the terminal does not persist in sending the packet and simply reschedules the transmission of the packet according to some random delay distribution. At this new point in time, the channel is sensed again and the same procedure is repeated.
- If a terminal learns that a packet collided with some other packet, it reattempts a retransmission according to the above procedure.

Defining by "S" the average number of packets generated per packet transmission time and by "G" the average number of new and rescheduled packets per packet transmission time it can be shown /17/ that "S" and "G" are related by:

$$S = \frac{G\,e^{-\delta G}}{G(1+2\delta) + e^{-\delta G}} \tag{3}$$

where "δ" the normalized propagation delay is given by:

$$\delta = \frac{\tau}{T_p} \tag{4}$$

where "τ" is the one-way propagation delay between any two terminals and "T_p" is the packet transmission time. It can also be shown /17/ that when a packet is ready for transmission, the probability "θ" that the channel is busy is given by:

$$\theta = \frac{G(1+\delta) - 1 + e^{-\delta G}}{G(1+2\delta) + e^{-\delta G}} \tag{5}$$

One of the characteristics of the NPCSMA random access scheme is that not all arrivals result in actual transmissions. Some packets will be blocked because of channel unavailability and only those that find the channel idle will actually be transmitted. Following the notation introduced in /17/ we will denote by "H" the actual rate of transmitted packets. Hence we can write:

$$H = G(1-\theta) \tag{6}$$

It is customary to define the probability of successful transmission by the ratio "S/G". However, since "G" includes packets that were not transmitted but were merely blocked prior to transmission, we believe it is more appropriate to define the probability of successful transmission by the ratio "S/H". From equations (3), (5) and (6) we obtain:

$$\xi = \frac{S}{H} = \frac{e^{-\delta G}}{1+\delta G} \tag{7}$$

For the case of voice packets we suggest a modification of the NPCSMA protocol, namely we do not attempt to retransmit a packet that has collided. This implies that, in the above equations, we should interpret "G" as the rate of new and rescheduled packets, "H" as the rate of transmitted packets and "S" as the rate of successfully transmitted packets.

SPEECH MODEL

In the previous section we have given expressions for the probability of sensing a busy channel and for the probability of successful packet transmission, in terms of "H" the normalized offered channel traffic. We must now relate "H" to the calling patterns of the voice sources as well as to their talkspurt statistical properties.

Consider a sequence of talkspurts and pauses characteristic of the speech pattern of a normal user and assume that the talkspurts as well as the pauses are exponentially distributed /18/ with means "$\bar{T}$" and "$\bar{P}$". Hence the probability density functions for the random variables "T" and "P" are given by:

$$f_T(t) = \frac{1}{\bar{T}}\ \exp\left(-\frac{t}{\bar{T}}\right) \tag{8}$$

and

$$f_P(t) = \frac{1}{\bar{P}}\ \exp\left(-\frac{t}{\bar{P}}\right) \tag{9}$$

We assume that the digitally encoded talkspurts are broken up into "n" packets of length "B". If the voice digitization rate is denoted by "R", we have:

$$P\,(RT \le nB) = 1 - \exp\left(-\frac{nB}{R\bar{T}}\right) \tag{10}$$

Hence, the probability that exactly "n" packets will be needed is:

$$P\,\{(n-1)B < RT \le nB\} = \exp\left(-\frac{(n-1)B}{R\bar{T}}\right) - \exp\left(-\frac{nB}{R\bar{T}}\right) \tag{11}$$

Thus the mean number of packets per digitally encoded talkspurt is obtained as:

$$\bar{N} = (1 - \exp(-\frac{B}{R\bar{T}}))^{-1} \approx \frac{R\bar{T}}{B} \quad (12)$$

It is well-known that the speech source activity ratio, "a", can be defined as the ratio of the average talkspurt duration to the sum of the average talkspurt duration and average pause duration, as follows:

$$a = \frac{\bar{T}}{\bar{T} + \bar{P}} \quad (13)$$

The source activity ratio can also be interpreted as the probability that an active speech source is issuing a talkspurt at some random time. Studies /18/ have shown that "a" is typically of the order of 0.4.

Since each packet generated during a talkspurt must be identified by a header of size "b" the time required to transmit a packet is given by:

$$T_p = \frac{B+b}{C} \quad (14)$$

where "C" denotes the channel transmission rate.

Finally denoting by "η" the offered load (in erlangs) per voice source during the busy hour, we obtain the following expression for the normalized offered traffic per source:

$$h = \eta\, a \frac{R}{C} \cdot \frac{B+b}{B} \quad (15)$$

Hence the total traffic offered to each of the "M" channels by a population of "N" voice sources is given by:

$$H = \eta\, a \frac{R}{C} \cdot \frac{B+b}{B} \cdot \frac{N}{M} \quad (16)$$

TRAFFIC MODEL

Consider Figure (2) where we see that during a talkspurt, packets "arrive" in a very regular manner. Indeed if each packet contains "B" bits, there will be a packet arrival every "B/R" units of time. This inter packet arrival time corresponds to the so-called packetization delay, that is the time required to form a packet of "B" bits.

Upon arrival of the first packet of a talkspurt to the buffer of the radio terminal, the process of selecting a channel for transmission, is initiated. Suppose that one of the "M" available channels is selected. After a period of time "t_s", the logic unit within the transceiver will decide whether or not the channel is busy. If that particular channel is found to be idle, a header is attached to the packet currently in the buffer and a packet of size "B+b" is transmitted. If the channel is sensed busy, another channel is selected at random among the available "M" channels and the process is repeated. Since the terminal has a buffer capable of containing a single packet and since the interarrival time of two consecutive packets is equal to B/R, a packet currently in the buffer that does not find an idle channel within this period of time is discarded.

More specifically if we divide the period of time $(B/R - T_p)$ into "k" slots each equal to the sensing time "t_s" we will discard a packet if after "k" sensing points an idle channel was not found.

Since, "θ", the probability that a given channel is busy, is the same for all channels, the probability "α" of being forced to discard the packet is given by:

$$\alpha = \theta^k \qquad k=1,2,\ldots \quad (17)$$

Naturally the probability (1-α) of being able to transmit the packet within the allowable time period is given by:

$$(1-\alpha) = \sum_{i=1}^{i=k} \theta^{i-1}(1-\theta) = 1-\theta^k \quad (18)$$

An important question can now be raised concerning the number of discarded and collided packets. Since we do not propose that collided packets be retransmitted, we would like to determine the fraction of lost packets (discarded and collided). A given packet within a talkspurt can be discarded with probability "α", and a transmitted packet can collide with probability "1-ξ", hence the fraction of lost packets in the uplink channel is given by:

$$\phi_u = \alpha + (1-\alpha)(1-\xi) \quad (19)$$

To determine the fraction of lost packets on the downlink channels we assume that packets arrive at the base station via "M" channels. Denoting by "S" the probability of a successful arrival over any one of these channels, if there are "L" downlink channels (L<M) the fraction of lost packets, "ϕ_d", is obtained from:

$$\phi_d = \frac{\binom{M}{L} S^L}{\sum_{i=0}^{L} \binom{M}{i} S^i} \quad (20)$$

To obtain the above expression we have assumed that no buffering is provided at the base station. This is a reasonable assumption since it is well-known that large delays in packet voice transmission will be intolerable. It should also be emphasized that so far we have delt with what we could call "incestuous" traffic. Indeed we have assumed that communications take place between mobile terminals, and have excluded communications coming in from or addressed to land terminals.

END-TO-END DELAY

For those packets that were successfully transmitted from origin to destination through the base station we can easily derive an upper bound for the maximum end-to-end delay as follows:

$$D = \frac{3B}{R} \quad (21)$$

where the first B/R is the packetization delay, the second B/R is the sum of the maximum pretransmission delay and the transmission time and the third B/R is the depacketization delay. Note that we have ignored in the above expression any processing delays that take place at the base station.

DISCUSSION AND RESULTS

Before applying the previous equations to a specific set of system parameters it is worthwhile to mention some of the factors that can play a role in selecting such parameters. Measurements carried out on the characteristics of speech during conversations have shown that human speech is bursty in nature. Brady /18/ among others has confirmed that the actual channel utilization during a one-way conversation is only about 40%. He has further shown that the exponential distribution fits reasonably well the distribution of talkspurt lengths with a mean value of about 1.3 sec. On the other hand, results /2, 19/ obtained to date on the transmission of packetized speech in the ARPANET indicate that to maintain a high quality speech it is necessary to ensure that:

- a nearly synchronous voice output is generated by the receiver
- end-to-end network delays do not exceed 250 msec.

Moreover, packets of lengths varying from 10 to 50 msec of speech intelligibility can be lost without seriously affecting the voice quality output and degradation begins to be observed when the fraction of lost packets exceeds a certain level, which is a function of the redundancy of the speech signal. According to some of the published data, the tolerable fraction of lost packets varies from 0.5% at low digitization rates to probably 50% at high digitization rates. Hence denoting by "φ" the total fraction of lost packets we have from equations (19) and (20):

$$\phi = \phi_u + \phi_d \leq \begin{cases} 0.005 \text{ for } R\approx 2.4 \text{ kbps} \\ 0.2 \text{ for } R\approx 16 \text{ kbps} \\ 0.5 \text{ for } R\approx 32 \text{ kbps} \end{cases} \quad (22)$$

From equation (16) it can be clearly seen that "H" the normalized offered traffic is highly dependent on the channel transmission rate "C". It is also clear that both the probability "α" of discarding a packet and the probability "1-ξ" of losing a packet, obtainable from Figures (3) and (4), are highly dependent on the values of "C" and "H". Hence by increasing the channel transmission rate we are clearly increasing the system efficiency measured in terms of the fraction of lost packets. There is however a practical upper bound on the transmission rate that can be derived from a 30 KHz land mobile channel. Indeed a number of modulation schemes that have been devised recently /20/, suggest that transmission speeds of the order of 40 kbps, can be achieved over a 30 KHz channel.

Another parameter that can influence the system performance is the packet overhead. For the purposes of our analysis we will assume that an abbreviated header of 16 bits /21/ is all that is required to properly address the packets.

Finally the last parameter of crucial importance is the time "t_s" required to sense a channel. As we have mentioned above, $(B/R - T_p)$ the period of time during which channels can be sensed is divided into a number of sensing slots of length "t_s". Hence the maximum number of sensing slots is given by:

$$K = \frac{B/R - T_p}{t_s} \quad (23)$$

which, for all practical purposes is a very large number. Indeed from Table 1, if we assume a channel speed of 40 kbps and a sensing time of 0.03 msec, we see that in the worst case "K" varies from about 53 (R=32 kbps and T_p=8.4 msec) to about 320 (R=32 kbps and T_p=40.4 msec). This immediately implies that for values of "θ" below 0.5, the probability "α" of discarding a packet can be ignored.

The values contained in Table 1 which were obtained for two temporal packet lengths of 10 msec and 50 msec (with η=0.02 and a=0.4) indicate, as expected, that for a given channel speed (40 kbps), as the voice digitization rate is increased, the average normalized offered traffic per voice source is also increased. This suggests that in order to increase the maximum number of voice sources that can be supported by the system, the voice digitization rate should be kept as low as possible. However, as indicated above the tolerable fraction of lost packets is a function of the speech redundancy which is quite low for low digitization rates. Since the tolerable lost packet level decreases faster than the normalized offered traffic per voice source, there is little advantage in decreasing the voice digitization rate beyond a certain value. It will be apparent that rather on the contrary, the voice digitization rate should be kept above 16 kbps. Consider a single radio channel supporting an amount of traffic "H" given by equation (16). If we assume that "ϕ_d", the fraction of packets lost on the downlink channel is negligible, the total fraction of lost packets when α → o, will be:

$$\phi \simeq \phi_u = \alpha + (1-\alpha)(1-\xi) \approx 1-\xi \quad (24)$$

Now from Figure (4) we see that, for δ = 0.1, "H" is related to "ξ" by:

$$H \leq \begin{cases} 1 \text{ for } \xi \geq 0.5 \\ 0.55 \text{ for } \xi \geq 0.8 \\ 0.03 \text{ for } \xi \geq 0.995 \end{cases} \quad (25)$$

which implies from (22) and the data of Table 1, that the maximum number of voice sources that can be supported is given by:

$$N \leq \begin{cases} \frac{1}{h} = 148 \text{ for } R=32 \text{ kbps} \\ \frac{0.55}{h} = 156 \text{ for } R=16 \text{ kbps} \\ \frac{0.03}{h} = 37 \text{ for } R=2.4 \text{ kbps} \end{cases} \quad (26)$$

Note that to derive the above numbers we have assumed a packet duration of 10 msec. For packet durations of 50 msec there is a slight increase in the number of users. However, from a delay point of view it is preferable as indicated by equation (21) to keep the packets as short as possible. Note also that in practice δ can be smaller than 0.1. If for instance we take a one-way propagation delay of 54 μsec and a packet transmission time of 1 msec we obtain δ=0.054. This implies in particular, that for R=2.4 kbps the maximum number of voice sources that can be supported exceeds the number given by equation (26).

In the case where we have "M" uplink channels, if we assume that the traffic is evenly distributed among these channels, the total number of voice sources that can be supported, is obtained by multiplying the results of equation (26)

by "M". Since we want to minimize the fraction of lost packets on the downlink channels, it is essential that for a given value of "S" we select the appropriate number "L" of downlink channels. As an example assume that the digitization rate "R" is equal to 16 kbps and that we can tolerate a total fraction of lost packets of the order of 20%. From Figure (4) (with δ=0.1) we find that "M" should be less than 0.55 hence the average successful traffic will be S=0.44. Using equation (20) with M=50, we see from Figure (5) that in order to keep "$\emptyset_d$" below 0.001, the number "L" of required downlink channels is of the order of 27. We can then achieve a spectrum saving of the order of 46% which for 30 KHz channels represents about 0.69 MHz. Additional savings in spectrum can be obtained by allowing "$\emptyset_d$" to increase while keeping "$\emptyset$" below 50%.

Various system configurations were simulated to determine the accuracy of "$\emptyset_d$" the theoretical fraction of lost packets on the downlink channels. In Figure (6) we see that the agreement between the theoretical and simulation results obtained for 4 uplink channels and 2 downlink channels, is excellent over a wide range of traffic loads.

CONCLUSION

Based on the results discussed above it appears that the concept of packet speech can be advantageously applied to land mobile channels. We have shown that for a given number of uplink channels, the maximum number of voice sources that can be supported will, under some assumptions attain a value of $\approx$ 150 M which is a threefold increase over what can be achieved with conventional analog land mobile channels. We have further shown that, as opposed to a conventional land mobile system where the same amount of spectrum is allocated in both directions, in a packet speech system the amount of spectrum required for the downlink channels represents about 50% of what is required in a comparable analog land mobile system.

ACKNOWLEDGEMENTS

The authors are thankful to G. van der Maas, H. Hafez and T. Kahwa for helpful comments on drafts of this paper and to G. Mousseau for his assistance in preparing the various curves. They are also endebted to D. Godin for her assistance in the typing of this paper.

REFERENCES

/1/ B. Gold, "Digital Speech Networks", Proceedings of the IEEE, Vol. 65, No. 12, pp 1636-1658, December 1977.

/2/ J.W. Forgie, "Speech Transmission in Packet-Switched Store-and-Forward Networks" Proceedings of the National Computer Conference, pp 137-142, 1975.

/3/ L. Kleinrock, S. Lam, "Packet Switching in a Multi-Access Broadcast Channel: Performance Evaluation" IEEE Trans. on Communications, Vol. COM-23, No. 4, pp 410-422, April 1975.

/4/ N. Abramson, "The ALOHA System - Another Alternative for Computer Communications" AFIPS Conference Proceedings, Vol. 37, pp 281-285, November 1970.

/5/ L. Schiff, "Traffic Capacity of Three Types of Common-User Mobile Radio Communication Systems" IEEE Trans. on Communications, Vol. COM-18, No. 1, pp 12-21, February 1970.

/6/ R.H. Frenkiel, "A High-Capacity Mobile Radio Telephone System Model Using a Co-ordinated Small-Zone Approach" IEEE Trans. on Vehicular Technology, Vol. VT-19, pp 173-177, May 1970.

/7/ S.L. Casner, E. Mader, R. Cole, "Some Initial Measurements of ARPANET Packet Voice Transmission" Proceedings of the National Telecommunications Conference, pp 12.2.1-12.2.5, December 1978.

/8/ I. Gitman, H. Frank, "Economic Analysis of Integrated Voice and Data Networks: A Case Study" Proceedings of the IEEE, Vol. 66, No. 11, pp 1549-1570, November 1978.

/9/ K. Bullington, J.M. Fraser, "Engineering Aspects of TASI" Bell System Technical Journal, Vol. 38, pp 353-364, March 1959.

/10/ S.J. Campanella, "Digital Speech Interpolation" COMSAT Technical Review, Vol. 6, No. 1, pp 127-158, Spring 1976.

/11/ S.S. Lam, "Packet-Switching in a Multi-Access Broadcast Channel with Application to Satellite Communications in a Computer Network" Ph.D. Dissertation, University of California, Los Angeles, Department of Computer Science, March 1974.

/12/ F. Tobagi, "Random Access Techniques for Data Transmission over Packet Switched Radio Networks" Ph.D. Dissertation, University of California, Los Angeles, Department of Computer Science, 1975.

/13/ J.B. deMercado, "An Overview of Packet Radio Communications" Paper presented at the Academy of Sciences, Moscow, May 1979.

/14/ Special Issue of the Bell System Technical Journal, Vol. 58, No. 1, January 1979.

/15/ Special Issue of the Review of the Electrical Communications Laboratories of Japan, Vol. 25, Nos. 11-12, Nov.-Dec. 1977.

/16/ Norway, "Signalling Channels in the Maritime Mobile Service" Contribution to IWP 8/5, September 1, 1978.

/17/ L. Kleinrock, F. Tobagi, "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and Their Throughput - Delay Characteristics" IEEE Trans. on Communications, Vol. COM-23, No. 12, pp 1400-1416, December 1975.

/18/ P.T. Brady, "A Statistical Analysis of ON-OFF Patterns in 16 Conversations" Bell System Technical Journal, Vol. 7, No. 1, pp 73-91, January 1968.

/19/ A.W. Huggins, "Effect of Lost Packets in Speech Intelligibility" NSC Note No. 78, 1976.

/20/ F. de Jager, C.B. Dekker, "Tamed Frequency Modulation - A Novel Method to Achieve Spectrum Economy in Digital Transmission" IEEE Trans. on Communications, Vol. COM-26, No. 5, May 1978.

/21/ D.W. Davis, D.L. Barber, "Communication Networks for Computers" New York, John Wiley and Sons, 1973.

PACKET SPEECH MODEL

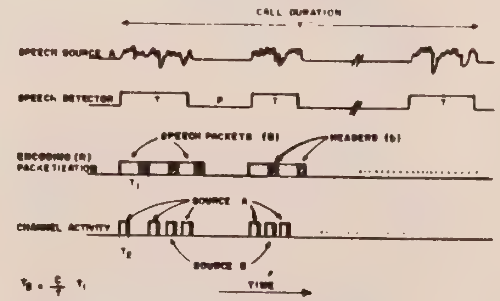
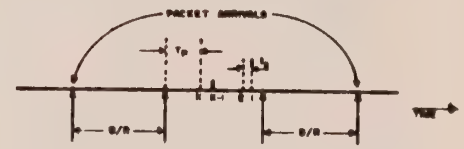


FIGURE 1

TRAFFIC MODEL



τ_s = SENSING PERIOD
T_p = PACKET TRANSMISSION TIME
D/R = TEMPORAL PACKET LENGTH

FIGURE 2

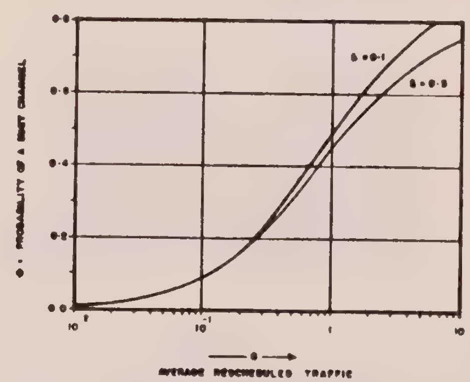


FIGURE 3 PROBABILITY OF A BUSY CHANNEL Φ AS A FUNCTION OF THE AVERAGE RESCHEDULED TRAFFIC G FOR TWO VALUES OF THE NORMALIZED PROPAGATION DELAY (δ)

R(kbps)	D/R = 10 msec D(bits)	D+h(bits)	T (msec)	h($\times 10^{-3}$)	D/R = 50 msec D(bits)	D+h(bits)	T (msec)	h($\times 10^{-3}$)
2.4	24	40	1	0.6	120	136	3.4	0.84
4.8	48	64	1.58	1.06	240	256	6.4	1.02
16	160	176	4.4	0.92	800	816	20.4	1.20
32	320	336	8.4	0.77	1600	1616	40.4	0.46

TABLE 1 AVERAGE NORMALIZED TRAFFIC (h) AS A FUNCTION OF TEMPORAL PACKET LENGTH (D/R) AND VOICE ENCODING RATE (R) FOR A CHANNEL SPEED OF 40 kbps

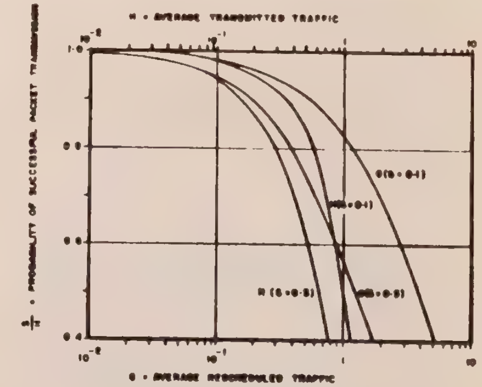


FIGURE 4 RELATIONSHIP BETWEEN TRANSMITTED TRAFFIC (M) AND RESCHEDULED TRAFFIC (G) FOR A GIVEN SUCCESSFUL TRANSMISSION PROBABILITY (1/T)

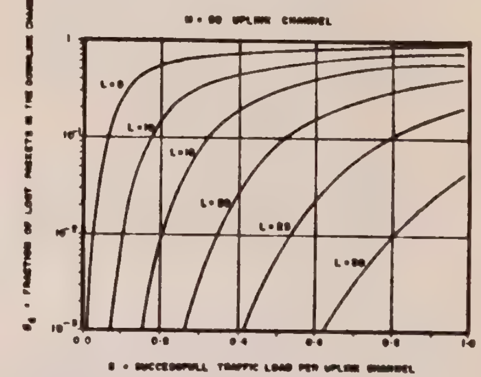


FIGURE 5 FRACTION OF LOST PACKETS ON THE DOWN LINK CHANNELS ($\emptyset_d$) AS A FUNCTION OF THE NUMBER OF DOWNLINK CHANNELS (L), FOR A GIVEN VALUE OF THE SUCCESSFUL TRAFFIC PER UPLINK CHANNEL (S)

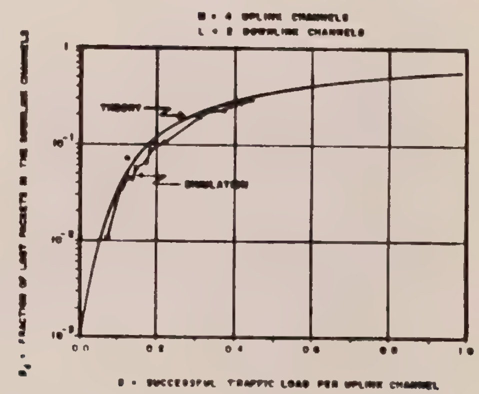


FIGURE 6 ACCURACY OF THE THEORETICAL MODEL

Reprinted from —
AFIPS — Conference Proceedings
Volume 44
© AFIPS PRESS
Montvale, N. J. 07645

ALOHA packet broadcasting—A retrospect

by R. BINDER,* N. ABRAMSON, F. KUO, A. OKINAKA and D. WAX
*University of Hawaii, THE ALOHA SYSTEM***
Honolulu, Hawaii

INTRODUCTION

Packet broadcasting is a technique whereby data is sent from one node in a net to another by attaching address information to the data to form a packet—typically from 30 to 1000 bits in length. The packet is then *broadcast* over a communication channel which is shared by a large number of nodes in the net; as the packet is received by these nodes the address is scanned and the packet is accepted by the proper addressee (or addressees) and ignored by the others. The physical communication channel employed by a packet broadcasting net can be a ground based radio channel, a satellite transponder or a cable.

Packet broadcasting networks can achieve the same efficiencies as packet switched networks,[1] but in addition they have special advantages for local distribution data networks[2] and for data networks using satellite channels.[3] In this paper we concentrate on those characteristics which are of interest for a local distribution data network. In particular, we discuss the lessons learned in the design and implementation of the ALOHANET, a packet broadcasting radio network in operation at the University of Hawaii since 1970. A number of design issues which arose in the construction of the system are defined, our solutions are explained, and in some cases they are justified. The lessons learned from the ALOHANET are used to indicate how such a radio packet broadcasting system might best be built using the technology available in 1975.

In the next section a brief description of the ALOHANET and its rationale is given. This is followed by a detailed discussion of the major system protocol choices that have evolved, pointing out some related theoretical work where appropriate. Choices concerning the design of the radio communication subsystem are then examined, followed by an evolutionary view of the important impact microcomputer technology has had on the user interface design and resulting system capabilities. The concluding section summarizes our present views with respect to the basic system configuration and properties of packet broadcasting nets.

* Now with Bolt Beranek and Newman, Inc., Cambridge, Massachusetts.
** Supported by the Advanced Research Projects Agency of the Department of Defense and monitored by NASA Ames Research Center under Contract No. NAS2-8590.

THE ALOHANET

The ALOHANET is the first system which successfully utilized the packet broadcasting concept for on-line access of a central computer via radio. Its primary purpose is to provide inexpensive access to one or more time-sharing systems by a large number of terminal users, typically in the hundreds. However, it also allows user-to-user communication within the net and is evolving toward use in a more generally-oriented computer communications environment.

Operation

The present network configuration makes use of a broadcast channel for only one direction of traffic flow. (As we shall see in later sections, the lack of a broadcast capability in the other direction has seriously handicapped the development of effective protocols in certain areas.) Two 100 KHz channels are used in the UHF band—a *random access* channel for user-to-computer communication at 407.350 MHz and a *broadcast channel* at 413.475 MHz for computer-to-user messages. The original system was configured as a star network, allowing only a central node to receive transmissions in the random access channel; all users received each transmission made by the central node in the broadcast channel. Recently the addition of ALOHA repeaters has generalized the network structure.

A block diagram of the present operational ALOHANET is shown in Figure 1. The central communications processor of the net is an HP 2100 minicomputer (32K of core, 16 bit words) called the MENEHUNE[4] (Hawaiian for IMP) which functions as a message multiplexor/concentrator in much the same way as an ARPANET IMP.[5] The MENEHUNE accepts messages from the UH central computer, an IBM System 360/65 running TSO (as of December 1974, a 370/158) or from ALOHA's own time-sharing computer, the BCC 500, or from any ARPANET computer linked to the MENEHUNE via the ALOHA TIP.[6] Outgoing messages in the MENEHUNE are converted into packets, the packets are queued on a first-in, first-out basis, and are then broadcast to the remote users at a data rate of 9600 baud.

The packet consists of a header (32 bits) and a header parity check word (16 bits), followed by up to 80 bytes of

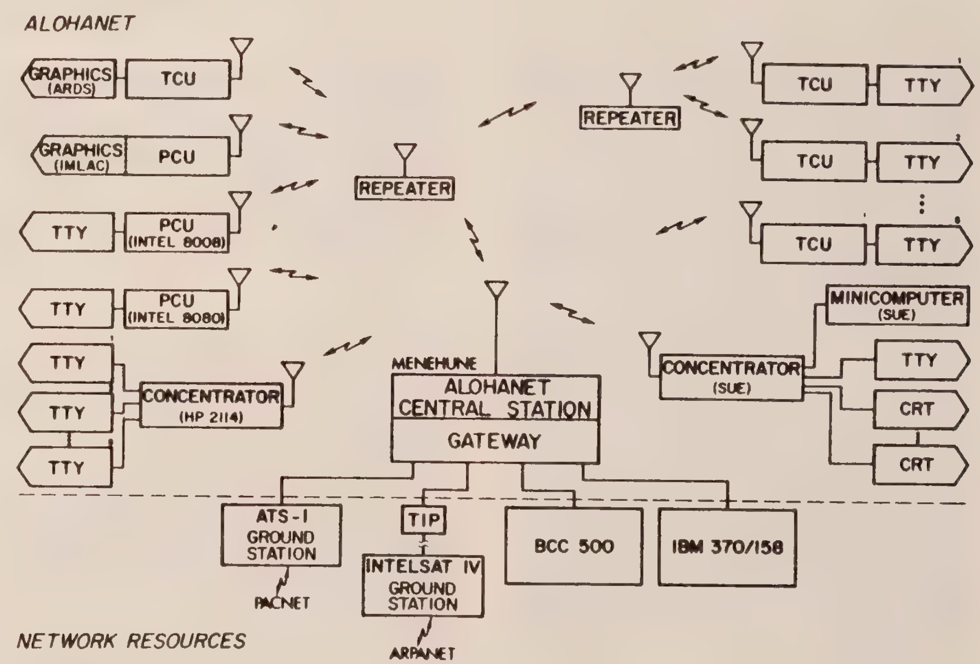


Figure 1 The ALOHANET

data and a 16-bit data parity check word. The header contains information identifying the particular user so that when the MENEHUNE broadcasts a packet, only the intended user's node will accept it. More will be said about packet formats later.

The random access channel (at 407.35 MHz) for communication between users and the MENEHUNE is designed specifically for the traffic characteristics of interactive computing. In a conventional communication system a user might be assigned a portion of the channel on either an FDMA or TDMA basis. Since it is well known that in time-sharing systems, computer and user data streams are bursty,[7] such fixed assignments are generally wasteful of bandwidth because of the high peak-to-average data rates that characterize the traffic. The multiplexing technique that is utilized by the ALOHANET is a purely random access packet switching method that has come to be known as the *pure ALOHA* technique.[8] Under a pure ALOHA mode of operation, packets are sent by the user nodes to the MENEHUNE in a completely unsynchronized manner—when a node is idle it uses none of the channel. Each full packet of 704 bits requires only 73 msecs at a rate of 9600 baud to transmit (neglecting propagation time).

The random or multi-access channel can be regarded as a resource which is shared among a large number of users in much the same way as a multiprocessor's memory is "shared". Each active user node is in contention with all other active users for the user of the MENEHUNE receiver. If two nodes transmit packets at the same time, a collision occurs and both packets are rejected. In the ALOHANET, a positive acknowledgment protocol is used for packets sent on the random-access channel. Whenever a node sends a packet it must receive an acknowledgment message (ACK) from the MENEHUNE within a certain time-out period. If the ACK is not received within this interval the node automatically retransmits the packet after a randomized delay to avoid further collisions. These collisions will limit the number of users and the amount of data which can be transmitted over the channel as loading is increased.

An analysis[9] of the random access method of transmitting packets in a pure ALOHA channel shows that the normalized theoretical capacity of such a channel is $\frac{1}{2}e = 0.184$. Thus the average data rate which can be supported is about one sixth the data rate which could be supported if we were able to synchronize the packets from each user in order to fill up the channel completely. Put another way, this result shows the present 9600 bit/second

channel could support between 100 and 500 active teletype users—depending upon the rate at which they generate packets and upon the packet lengths.

ALOHANET remote units

The original user interface developed for the system is an all-hardware unit called an ALOHANET Terminal Control Unit (TCU), and is the sole piece of equipment necessary to connect any terminal or minicomputer into the ALOHA channel. As such it takes the place of two dedicated modems for each user, a dial-up connection and a multiplexor port usually used for computer networks. The TCU is composed of a UHF antenna, transceiver, modem, buffer and control unit.

The buffer and control unit functions of the TCU can also be handled by a minicomputer or a microcomputer. In the present system several minicomputers have been connected in this manner in order to act as multiplexors for terminal clusters or as computing stations with network access for resource sharing. A new version of the TCU using an Intel 8080 microcomputer for buffer and control has been built. Since these programmable units allow a high degree of flexibility for packet formats and system protocols, they are referred to as PCU's (Programmable Control Unit). A more detailed discussion of terminal considerations is given in a companion paper in these proceedings.[9]

Since the transmission scheme of the ALOHANET is by line-of-sight, the radio range of the transceivers is severely limited by the diversity of terrain (mountains, high rise buildings, heavy foliage) that exists in Hawaii. A recent development has allowed the system to expand its geographical coverage beyond the range of its central transmitting station. Because of the burst nature of the transmissions in the ALOHA channel it is possible to build a simple store-and-forward repeater which accepts a packet within a certain range of ID's and then repeats the packet on the same frequency. Each repeater performs identically and independently for packets directed either to or from the MENEHUNE. Two of the repeaters have been built which extend coverage of the ALOHANET from the island of Oahu to other islands in the Hawaiian chain. These repeaters are discussed in more detail in the following section.

PROTOCOL CHOICES

Two fundamental choices which have dictated much of the system protocol are the two-channel star configuration of the original network and the use of random accessing for user transmissions. Investigation of the random accessing principle using radio was in fact the original motivation for constructing the ALOHANET, while the two-channel configuration was primarily chosen to allow this investigation without complication from the relatively dense total traffic stream being returned to all users. An additional reason for the star configuration was the desire to centralize as many communication functions as possible at the MENEHUNE, minimizing the cost of the TCU at each user node.

Within this context, a number of protocol issues must be resolved. The more important of these are:

- random access channel control
- broadcast channel queueing
- packet length
- addressing
- error control
- flow control

Many of the original choices in these areas have undergone significant changes as a result of new user resources and user interfaces, or in some instances due to advancements in theoretical knowledge. The addition of repeaters has (potentially) a particularly significant impact on protocol.

We now discuss some of the considerations and resulting choices made in each of the above areas, with the impacts of new factors introduced within the context of each area. The section concludes with a brief discussion of the problem of integrating file traffic into the random access channel, a subject of current concern in the ALOHANET.

Random access channel control

The retransmission strategy used in the random access scheme plays a central role in the scheme's effectiveness. Its determination directly affects the average delay experienced by users for a successful transmission, given a certain number of users accessing the channel, their traffic statistics, and the channel capacity. It can also be used to prevent the occurrence of channel saturation, a situation in which the channel becomes filled with retransmissions and the number of successful packets falls to zero. These topics have only recently been quantified[10,11] and remain subjects of current investigation.

One approach is to use different constant retransmission intervals at each node, with the intervals equal to integer multiples of the maximum packet transmission time to avoid subsequent conflicts. This results in a priority structure, since nodes assigned the longer intervals will experience a correspondingly longer average delay. As the number of nodes becomes large, however, unacceptably large delays result for the majority of users.

A strategy more appropriate for large user populations is to randomize the retransmission intervals used at each node (note that a priority structure can still be introduced if desired by using larger mean values for lower priority users—in the remaining discussion, equal priorities will be assumed). According to recent results by Lam,[11] the resulting channel behavior appears to be relatively insensitive to the exact nature of the randomization, at least when comparing the use of uniform and geometric distributions. In any event, the cost of implementing a particular distribution at each node is an important design consideration.

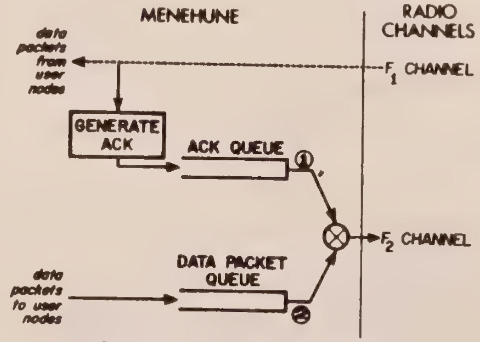


Figure 2—Broadcast Channel Multiplexing

Based on initial estimates of the expected ALOHANET characteristics, a choice was made to use a uniform distribution. This allowed a relatively simple implementation in both hardware and software user nodes.

A simple technique was used in the original system nodes to achieve short delays when the channel is lightly loaded, while preventing channel saturation from occurring due to peak-hour loading or statistical traffic fluctuations: small retransmission intervals are used (relative to the intervals between new packets), but only for a maximum of three successive retransmission attempts. If the third attempt is unsuccessful, the user is notified of a failure and must manually reinitiate the retransmissions. This in effect introduces a long interval between every three retransmissions, allowing time for retransmissions from other users to succeed. Based on a maximum packet transmission time of 70 milliseconds, the intervals are selected from a range of 0.2 to 1.5 seconds, giving a mean of about 0.7 seconds (ten maximum packet times) per retransmission. The lower bound is chosen to allow sufficient time to receive an ACK from the MENEHUNE if the packet was sent successfully, avoiding unnecessary retransmissions. (This time is based on a direct user-MENEHUNE path; if repeaters form a part of the radio path, the lower limit must be increased accordingly.)

The newer programmable PCU's in the system offer the capability of a more flexible strategy, for example allowing the interval used after each third retransmission to be automatically inserted. The use of different strategies, such as continuously increasing the time range used for selection of successive retransmissions, is also easily implemented by program; these and other strategies are currently under investigation.

Broadcast channel queueing

The MENEHUNE acts as a concentrator for the broadcast (F_2) channel, queueing waiting traffic when necessary for sequential transmission to user nodes. Four complicating factors exist, however: a need for priority queueing, fair allocation of the channel, the turnaround delay required by half duplex nodes, and the presence of repeaters.

Priority queues

It is important that the F_2 channel data traffic not prevent the prompt return of an ACK to a user node, since this could lead to unnecessary user retransmissions and possible degradation of the random access (F_1) channel. Thus, an integral part of the F_2 channel multiplexing is the priority queueing mechanism maintained by the MENEHUNE, as shown in Figure 2. Whenever a transmission is completed on the F_2 channel the ACK queue is checked, and if not empty the ACK at the head of the queue is sent. Only when the ACK queue is empty is the data packet queue checked for waiting packets. This guarantees that at most one complete data packet plus any previously queued ACK's will be sent ahead of an ACK just placed on the queue. (Because the average rate of successful arrivals on the F_1 channel is limited to one-sixth the rate of F_2 transmissions by the random access technique, the number of previously queued ACK's will be zero most of the time.)

Fairness

A second problem is the possible hogging of the F_2 channel by one or a few users. This problem is eliminated by the queueing discipline used for the data packet queue. Only one packet per user is allowed on the queue at any time, and the queue is serviced on a first-come-first-served (FIFO) basis. The prevention of more than one packet per user on the queue is handled in conjunction with user flow control, discussed below.

Turnaround delay

A delay function is used by the MENEHUNE to count off the time required by half-duplex user nodes to switch from a transmit to a receive state. The actual time is determined by the equipment type—the original off-the-shelf equipment required 100 milliseconds due to its use of mechanical relays; approximately 10 milliseconds is counted off for newer equipment now in use.

Repeater scheduling

The addition of repeaters to the system introduces a number of new problems into the F_2 channel, both because of radio range overlap and the nature of the repeaters themselves. The latter are store-and-forward devices; a packet which is to be repeated is first received and stored in its entirety, then transmitted on the same frequency on which it was received (preventing reception of a new packet during this time). In order to prevent the loss of a

second packet destined to the same repeater, the MENEHUNE must therefore appropriately schedule the packets in its F_1 channel queues.

For efficient scheduling (i.e., to maximize channel utilization), the MENEHUNE must know the repeater routing paths for each user node. This function could thus become quite complicated or even not achievable, depending on the degree of dynamic routing used. Because of the small percentage of traffic currently handled by repeaters in the present ALOHANET, a very simple brute force method is used: whenever a packet is sent which is forwarded by one or more repeaters, the MENEHUNE counts off sufficient time for it to be repeated once before beginning a new transmission to any node (knowledge of which packets are to be repeated is available from the user address, discussed below). This results in wasted channel capacity, but is not significant due to the capacity available in the system at present.

Packet length

Three factors having an important impact on the system are the use of variable or fixed-length packets, the way packet length or the number of data bytes is indicated, and the maximum packet length allowed. The choices made must take into account the different traffic characteristics generated by line-oriented and character-oriented user-computer interactions.

Line transmissions

Fixed-length packets were used in the initial system to simplify the design and construction of system hardware. The data packet length for both channels was chosen to allow up to 80 data bytes (640 bits), based on the user delays introduced by the 9600 bps channel data rates, the line length of the terminals in the system, and the line-oriented characteristics of the IBM 360/65 used as the central time-sharing system. An end-of-line (EOL) indicator consisting of eight zero bits was used within the packet to identify the end of actual data, where the latter was restricted to 7-bit ASCII with the eighth (parity) bit set to one. Since it was anticipated that many of the lines typed by users would be less than 40 characters, a second packet type was also defined which contained a 40-byte data field (a "Half-Packet"). This last step proved to be a mistake—the half-packet logic at each end of the link was a significant source of both hardware and software bugs.

The packet formats have since been changed to allow the use of variable-length packets with newer user nodes. An 8-bit count field is used in the packet header to indicate the number of 8-bit data bytes in the packet, with the data parity word immediately following the last data byte. In addition to eliminating the wasted channel capacity of the fixed-length packets, this also removes constraints on the data themselves necessitated by unambiguous detection of the EOL indicator within the data stream. The 80 data-byte maximum has been retained for both channels, since it still appears to be a reasonable upper bound with respect to both the multiplexing delays introduced to either channel and node buffering requirements. This should not be construed as an indication that this length is optimal, however; as file-oriented messages are introduced to the total traffic and/or user node storage continues to become cheaper, a larger maximum may be desirable for one or both channels (for a given channel data rate and user response time constraints).

Character-by-character

The increased flexibility provided by PCU's has allowed the introduction of a 'short' data packet in which a single data byte is sent in the header in place of the byte count, followed only by the header parity word. Although a use for this packet occasionally arises for interactions with line-at-a-time systems, its main use is with the character-oriented ARPANET computers now available to ALOHANET users.

The use of these character-oriented systems can have a considerable impact on the size and frequency of packets sent in the random access channel. This has an important consequence for the buffering strategy and choice of packet length used at each node: since a new transmission cannot begin until an ACK has been received for the last one, all characters typed by the user during the ACK waiting time should be sent in a single packet. Thus if communication delays tend to overlap inter-character generation times, the affected characters are accumulated at the originating node and sent (more efficiently) in a variable-length packet, without adversely affecting user-computer interaction.

A logical extension of this last strategy is to buffer all characters typed by the user at his node until one is typed which causes some action to be taken by the computer. If the appropriate set of action characters is known at the user node, this allows an optimum use of both channel capacity and system buffering without degrading the user-computer interaction. A scheme which allows this to be done in conjunction with echoing control is given by Davidson,[11] and is currently being introduced into selected ARPANET hosts. Its implementation cost in ALOHANET PCU user nodes appears reasonable, and is anticipated for use as its support by host computers becomes widespread.

Addressing

User nodes

User addressing is determined by the radio channel configuration and associated multiplexing technique. Ignoring repeaters for the moment, the two-frequency configuration used in the ALOHANET allows only a single destination in the random access channel (the MENEHUNE), and a single source in the broadcast channel (the MENEHUNE). Thus only the sender's address is required in the random access channel and only the destination address in the broadcast channel, which in both cases is the user address. Concentration of more than one user at a radio node is handled by permanently allocating a block of user addresses to the node, allowing user node multiplexing without introducing another level of addressing complexity to the system. The required address space is determined by the total number of users expected to be supported by the random access channel, and is 2^8 (eight header bits) for the present 9600 bps ALOHANET channel.

Repeaters

The use of repeaters in the system introduces some significant new factors to be considered in choosing an address scheme. Because of radio range overlap and the store-and-forward nature of the repeaters, problems can arise involving conflicts generated by two or more repeaters repeating simultaneously to the same destination, infinite repeating of the same packet (looping), and weak-signal operation due to multiple (but time-sequential) paths. In addition, the addressing scheme directly affects the MENEHUNE's ability to schedule transmissions in order to maximize broadcast channel utilization, as discussed in a preceding section. The ability to eliminate or minimize these problems depends on the degree of mobility desired for user nodes and/or the repeaters themselves.

Because of the small percentage of user nodes which currently require repeaters in the ALOHANET, a simple scheme is in use based on the hardwired properties of the original repeaters built for the system. A block of user addresses is defined for each repeater, the latter repeating only those addresses in its block. The block assigned to a repeater two hops from the MENEHUNE is a subset of the block assigned to its first hop repeater. User nodes are constrained to operate within the geographic range of their 'assigned' repeater by this scheme, but the node's user address is easily changeable if a relocation becomes necessary. Since only one path choice exists between each user node and the MENEHUNE at present, the optimum path is selected by default. As the number of repeaters in use increases and existing units are replaced by programmable devices, a more flexible repeater addressing scheme is expected to be implemented.

Resource addressing

This refers to the user's choices regarding which system resource he may communicate with. The system allows users to request a connection to the campus IBM 370/158, the ARPANET, or another ALOHANET user node. This is accomplished by sending special sequences of ASCII characters in the data portion of packets to the MENEHUNE, which may either be typed by a terminal user or automatically generated. If the requested destination is available, its identification is stored in a *Connection Table* entry for the requesting user in the MENEHUNE, and the user's address stored in a similar entry for the destination. All subsequent packets from the user are passed to the stored destination and conversely, until either end requests that the connection be broken.

Two exceptions exist to this connection table routing of packets. The first are commands intended for the MENEHUNE, such as the 'connect' and 'disconnect' above. The second is a capability which allows a user to send a single packet to another ALOHANET user independently of current connection table entries. The originating user simply types a special two-character ASCII sequence followed by the destination user's address (up to three ASCII digits), followed by the desired text.

Note that in the case of a connection to another ALOHANET node, the latter's address is also the resource address. If the node's resource can service more than one user at a time (such as might be the case for a specialized minicomputer or storage device), the present addressing scheme requires either that a block of addresses be allocated to the receiving node (as in the case of a concentrator for sending), or a sub-address be sent in the text portion of every packet. The block allocation suffers from rigidity in that resource addresses cannot be reused dynamically by different users, and does not appear desirable if many such addresses must be allocated in the system.

Error control

Random-access channel

Two distinct error sources exist at the MENEHUNE receiver, the usual random noise and errors due to packet conflicts. Because of the high probability of errors due to conflicts at full loading of the random access channel, a very reliable error detection mechanism is required. To achieve this it was decided to use two 16-bit cyclic polynomial parity check words in each data packet, one following the header and a second following the data. The separate header parity check forms the basis for a highly reliable packet synchronization method discussed in another part of this paper; it also allows reliable establishment of packet length and other information prior to processing the data portion of a packet. A single header bit is also used in conjunction with the parity check for sequence numbering, allowing the detection of duplicate packets by the MENEHUNE.

Broadcast channel

Error control for broadcast channel data packets (MENEHUNE to user nodes) involves some special considerations. For efficient operation, the usual positive acknowledgment scheme in which the ACK's themselves are not acknowledged depends on a high probability of the ACK's being successfully received. However, an ACK sent

from user nodes must compete with data traffic in the random access channel. At full channel loading each random access packet must be retransmitted an average of 1.7 times, which means each data packet or ACK must be sent a total of 2.7 times on the average before it is successfully received.* But in order to force retransmission of the ACK's, the data packet being acknowledged must also be sent an average of 2.7 times by the MENEHUNE—even though it was received correctly the first time! The problem is compounded by the typically high ratios of computer/user traffic which exist for most interactive systems, resulting in many more ACK's than data packets in the random access channel. This problem was "resolved" for the initial implementation by simply not sending ACK's from user nodes. Because of the high received signal strengths at the nodes, a very low error rate was anticipated; considering also that user nodes consisted only of human terminal users, it was decided that a simple error detection/user notification scheme would be sufficient.

However, this is in general not adequate when more sophisticated data transfer functions take place or significant error rates exist at user nodes. An example of the first case is the loading of programs into core storage of a minicomputer node, where manually initiated error recovery usually requires restarting the loading from the beginning of the file. In the second case, error rates can become appreciable when user nodes are located in weak signal areas caused by distance, multipath interference, or line-of-sight blocking, or in strong signal areas in which strong local noise sources also exist. To allow for these situations, an option which allows user nodes to send positive acknowledgments has been implemented. The scheme works identically to that for the random access channel, but is only used selectively with newer programmable nodes when required (it can be turned on or off by a command from the user node to the MENEHUNE). Its effectiveness is based on the relatively light existing channel loading of the system and its use by only a few of the nodes.

One solution to this problem when all traffic to user nodes must be acknowledged in a loaded random access channel is to use sequence numbering with a large modulus, sending an ACK only when the maximum sequence number is received. This approach suffers from the unpredictable nature of interactive user-computer traffic, however; if the last computer output prior to new user input is missed by the node, a potential deadlock situation is created until the user decides something is wrong and takes manual action. An additional mechanism can be used to circumvent this, such as using automatic timeouts at the user node or sending dummy traffic to the node to 'flush out' missed packets. However, the sequence numbers succeed only in reducing the number of ACK's sent in the random access channel—to eliminate the unnecessary repetitions of data packets from the MENEHUNE, it is also necessary to acknowledge the ACK. That is, the ACK sent by a user node is timed out and retransmitted until an acknowledgment for it is received, just as for data packets. If another packet is waiting for transmission to the node at this time, its transmission with the next sequence number constitutes the ACK to the ACK; otherwise, a short ACK-ACK packet is sent by the MENEHUNE. This can be easily shown to result in significantly less total channel overhead, at the expense of more complication in the node implementation.

Repeaters

We have so far ignored the effects of repeaters in this discussion on both random access and broadcast channel error control. The repeaters currently in use in the ALOHANET do not generate acknowledgments in either direction, resulting in only end-to-end acknowledgments between the MENEHUNE and user nodes as above (but with longer minimum retransmission timeouts). This choice was made for initial repeater simplicity; it has been shown analytically, however, that a hop-by-hop acknowledgment scheme is in general superior to an end-to-end scheme, at least in contexts such as ARPANET[18] and the ARPA Packet Radio effort.[19] Thus we expect to convert to a hop-by-hop scheme when the existing repeaters are replaced by programmable units and/or repeater traffic error rates require it; this area remains a relatively unexplored problem domain within the present ALOHANET implementation.

Single-channel configurations

Finally, we note that the problems discussed above concerning ACK's sent by user nodes in the random access channel are effectively non-existent if a single-frequency channel configuration is used (and propagation times are less than the shortest packet transmission times). If all nodes can hear the transmission of all other nodes, it is only necessary that nodes refrain from sending for an ACK packet time following the transmission of a data packet by any node, except for the intended receiver who sends an ACK (if appropriate) during this time. Thus ACK's are sent conflict-free, allowing a simple positive acknowledgment scheme to be used for all traffic. Note that packets sent by the MENEHUNE are treated exactly the same as packets sent by user nodes with respect to ACK's, thus also eliminating any effects due to asymmetric computer-user traffic ratios.

Flow control

The initial system

In the initial system environment of a single half-duplex time-sharing system, model 33 Teletypes, and hardwired

---

* This assumes ACK's and data packets are the same length, although the ACK's are in fact shorter, the resulting error rate is still very high compared to a typical conflict free channel

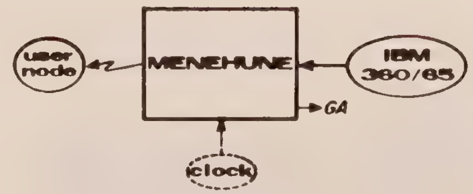


Figure 3 - Broadcast Channel Flow Control (Original System)

user nodes which buffered only the line being displayed, flow control was a relatively simple matter. A user always received at least one output line from the time-sharing system (IBM's TSO running on a 360/65) for each input line, and a prompt character when it was ready for more input. The bandwidth between the MENEHUNE and 360 and the latter's I/O response times are such that one or two MENEHUNE buffers are normally sufficient to support transfers of packets received from the random access channel; in the unlikely event that no buffers are available when a packet arrives, the channel protocol guarantees its retransmission. Thus no explicit flow control was provided to prevent new packets from being sent by a user node. If the user sends one before the 360 is ready, the packet is discarded and a "WAIT" message returned to the user by the MENEHUNE (the status of each 360 connection is known in the MENEHUNE by information routinely passed from the 360).

Broadcast channel flow control was necessary, however, since each line (packet) sent to a (hardwired) user node must be completely displayed before a new line can be received. This was accomplished by the scheme shown in Figure 3, in which the control for each user node is centralized at the MENEHUNE. The latter counts off the required display time following transmission of each packet to a user, inhibiting further transmissions to that user until the time is up. To prevent 360 output from tying up MENEHUNE buffers while packets are being displayed, a handshaking flow control is used; the 360 sends only one line of output for each user, then waits for a *go-ahead* (GA) message with that user's address. The GA is sent by the MENEHUNE whenever a user's display time is up, resulting in at most one buffer required for each user (the MENEHUNE can also hold up acceptance of any packet from the 360 indefinitely until it has buffer space available). Note that this strategy also prevents any user from hogging the broadcast channel, since it allows only one packet per user in the channel queue.

Some terminal complications

The introduction of high speed CRT and hardcopy terminals to the system required an expansion of the MENEHUNE's flow control mechanism for the broadcast channel. A set of display rates was added, with the rate used at each user node stored in a permanent table in the MENEHUNE; a user can change the stored value for his node by typing a special command to the MENEHUNE at any time. The CRT terminals require an additional flow control mechanism to suspend output when the CRT screen has filled, allowing the user to signal when he is ready to proceed. Thus a screensize command was created which allows users to specify a screensize of between one and 99 lines (or an infinite screensize); this value is also stored in MENEHUNE tables for each user node. A counter is maintained for each user with a finite screensize specification and is updated for each line sent to the terminal; when the maximum is reached, the MENEHUNE suspends generation of the GA message until the user inputs a carriage return.

Satellite complications

The next complication to MENEHUNE flow control processing was caused by the connection of the ALOHANET to the ARPANET. The latter involves a 50 Kbps INTELSAT IV satellite path connecting Hawaii to California; because of its long propagation time (approximately 0.25 seconds) and ARPANET flow control protocol, a large amount of buffering is required at the receive end of the link to support continuous display at higher speed terminals—in particular, a 9600 bps terminal requires approximately a 1000-byte buffer. (Since in general CRT terminal users do not require continuous output at this rate, a smaller amount of buffering is in fact used.) This required a substantial increase in the size of the MENEHUNE buffer pool and a more complicated queueing structure to support the broadcast channel, since now more than one packet per user must in general be stored in the MENEHUNE during display at the user node. To maintain the single-packet-per-user policy for the channel queue, a separate queue was created for each user to hold additional packets. The resulting flow control scheme is shown in Figure 4, where the GA's sent to the 360 in Figure 3 are now sent to the internal ARPANET protocol module. The maximum allowed size of each user queue is determined by the user's terminal rate and the available MENEHUNE buffer pool, and in turn defines

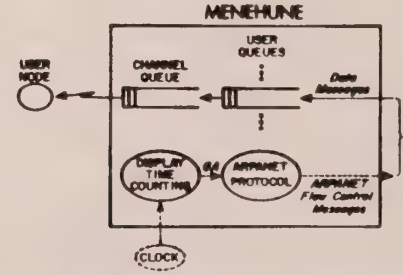


Figure 4 - Broadcast Channel/ARPANET Flow Control

the parameters used in the ARPANET flow control protocol.

Multiple-line packets

A second complication resulting from the ARPANET connection concerns the extra time required by some higher speed displays for certain characters such as carriage return (CR) and/or line feed (LF). Output from the 360 in the initial system contained such characters only at the end of a line (packet), allowing the transmission time and other inter-packet delays to provide any extra time required. However, many ARPANET computers are character-oriented, at times generating many CR and LF characters within a single packet. Thus it was necessary to provide a padding function in the MENEHUNE which inserts dummy characters or otherwise adds a display time delay after each CR or LF occurrence within packets destined for a higher speed (greater than 110 bps) terminal. This necessitates the splitting of packets whenever the maximum 80-byte packet length is exceeded, and in general involves a significant amount of additional processing per packet.

Full duplex interaction

A third complication arising from many ARPANET computers is their full duplex user interaction. Unlike the 360, users do not necessarily receive output in response to each input or an indication of when the computer is waiting for more input. Since no explicit flow control is provided for input from user nodes to the MENEHUNE, users are forced to either interact in a half duplex fashion (guessing as to when the computer has finished its output) or suffer occasional losses of input data and subsequent retyping. The latter can occur frequently with the hardwired TCU's, since they contain a single buffer which is used for both keyboard input and display; if computer output arrives while the user is typing, the typed characters are overwritten in the buffer by the received packet. The newer programmable user nodes now in the system provide full duplex buffering for the terminal, allowing a packet to be received and displayed without disturbing the keyboard buffer.

However, even if user nodes are completely full duplex a flow control problem exists for packets sent to the MENEHUNE. Unlike the case for the 360, users of full duplex hosts may generate successive input packets without receiving responses from the host computer. If the ARPANET or host computer or both slow down, an excessive number of buffers can become queued in the MENEHUNE on behalf of the user. Thus, to prevent user hogging of the buffer pool a count of the number of input buffers queued for each user is now maintained; when equal to the maximum allowed, arriving packets are discarded and a discard notification returned to the user.

File traffic

The original ALOHANET design was based on a homogeneous population of terminal users generating bursty traffic into the random access channel. However, the connection of minicomputers and other terminals with memory has introduced at least two sources of non-bursty, or 'file', traffic. The first case occurs when users desire to transfer data from a paper tape or other storage media to a host computer. The second occurs when it is desired to transfer program-generated output from a minicomputer at a user node to a display device at a second user node (users can connect to other user nodes through the MENEHUNE in the same way as to the 360 or ARPANET). In either case the resulting traffic must be prevented from hogging or degrading the random access channel, and must also be constrained to the destination's acceptance rate.

The random access technique itself implicitly provides an anti-hogging mechanism, since retransmission timeouts can be used to decrease the user's average rate if conflicts occur. This does not provide for destination flow control, however, and is not necessarily an optimal solution for the random access channel. A second approach is the use of explicit flow control in the form of GA's sent by the MENEHUNE to the sending user node. This provides a solution to both problems at the expense of a small percentage of broadcast channel capacity. Since the MENEHUNE receives GA's from the user's destination, either explicity from the 360 or ARPANET module or from its display time counting for another ALOHANET node, it can simply relay them to the sending node in a short control packet. This approach also allows centralized optimization of traffic in the random access channel by the MENEHUNE, and is the subject of current investigation.

RADIO SUBSYSTEM CHOICES

The design of the ALOHANET radio communication system required the balancing of a number of performance goals against various system constraints which are peculiar to the use of radio frequencies for data communication channels. These trade-off studies resulted in the selection of our RF channels and modulation method. The determination of operating ranges and the choice of a data synchronization method resulted from the basic channel and modulation selection decisions. In this section we will describe the primary issues related to RF channel selection, modulation design, radio range determination, and data synchronization design.

RF channels and modulation

The choice of radio channels for any communication system is a complex task, requiring the trade-off of many factors such as desired bandwidth, area coverage, spectrum availability, potential interference and noise

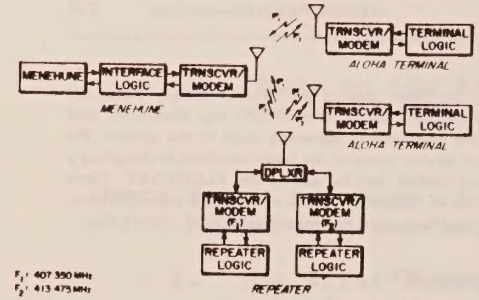


Figure 5—ALOHA System UHF Radio Communication System

sources, regulatory requirements, and equipment costs. In the case of the ALOHANET, a wide channel bandwidth was considered desirable for the random access channel since user nodes are required to send messages to the MENEHUNE at high peak data rates compared to their average data rate. Wide bandwidth was also deemed advisable for the broadcast channel due to the expected high traffic density from the MENEHUNE. The use of wide channel bandwidth tends to force the use of higher frequencies where spectrum crowding is less severe and the availability of bandwidth is greater. Crowded radio bands are undesirable not only from the standpoint of interference to other users but also because of potential interference from them. Another disadvantage of lower frequencies is the higher probability of interference from man-made noise sources, particularly in an urban area where the ALOHANET has most of its terminals.

From the above considerations it can be seen that the system's communication requirements tend to emphasize the use of higher radio frequencies. The primary constraint on moving to even higher frequencies is equipment cost and radio range. Above 500 MHz equipment costs tend to escalate rapidly. Area coverage also becomes more difficult due to more pronounced shadowing effects of the radio waves by buildings and hilly terrain. (Above 30 MHz radio propagation tends to be limited to line-of-sight paths.)

Therefore, the 400 to 500 MHz UHF band was selected as the optimum for the ALOHANET radio frequencies. Reasonably priced commercial radio equipment was found to be available in this frequency region and radio band crowding was not severe in Hawaii. Initially, assignments in the 450 to 470 MHz mobile radio band were requested but were rejected by the FCC because of our wide channel bandwidth requirements. (The mobile radio channels are specified at about 15 KHz bandwidth, whereas we were requesting 100 KHz.) We were fortunate enough to receive assignments as an experimental service in the government UHF band of 406 to 420 MHz, where spectrum space was available.

Since most radio equipments available in the UHF bands use frequency modulation (FM), this type of modulation was selected for the RF channels. A slight variation was incorporated in the hardware design to minimize the interface problems between the radios and the data modems. This variation was the use of a subcarrier tone to carry the actual data modulation. This tone is phase-shift-keyed by the data and the resultant signal is used to modulate the FM transmitter. This modulated tone is recovered from the FM receiver and fed to the demodulator of the modem. This modulation system is referred to as FM/DPSK to indicate frequency modulation by a differentially phase-shift-keyed subcarrier. (Differential phase-shift-keying is used to resolve the problem of received phase ambiguity.) The resultant configuration is shown in Figure 5.

Radio range

The maximum operating distance between any terminal of the ALOHANET and the MENEHUNE (or a repeater) is specified as the system's radio range. This distance is primarily a function of a transmitter's radiated power, the receiver's sensitivity, and the attenuation of radio signal power for the given distance. Local noise conditions at the receiver location can also affect this distance, but for system planning purposes, range is usually calculated on the basis of some given propagation model. For line-of-sight paths, which exist at VHF, UHF, and higher frequencies, two different models are used depending upon local topographical conditions. In an urban area these paths are partially obstructed and suffer from multipath effects. A power loss proportional to $1/R^4$ is usually assumed for these conditions.[14] Where paths are unobstructed and well clear of the local terrain, a spreading loss proportional to $1/R^2$ can be assumed. Receiver threshold sensitivity in the ALOHANET is defined as that receiver input power level which causes an average bit error rate of 10^{-5}. This bit error rate should provide a packet throughput reliability better than 99 percent for full-length ALOHA packets.

Assuming a transmitter equivalent radiated power of 10 watts, a simple whip antenna at a user terminal, an elevated antenna at the MENEHUNE or repeater and a 3 microvolt receiver sensitivity, the radio range works out to about 17 miles in the urban area for the ALOHANET frequencies. Between repeaters and the MENEHUNE terminal, which have well-elevated antennae and good path clearances, the assumed $1/R^2$ model gives a maximum range of 290 miles. The use of high-gain omnidirectional antenna arrays at repeater sites extends these ranges. Tests conducted on a 100 mile path between two ALOHANET repeaters confirmed the $1/R^2$ spreading-loss assumption and indicated a fade margin of 30 db existed (due to the 10 db gain antennae used for the test).

*Data synchronization**

Because of the burst nature of radio transmission of ALOHANET packets, special synchronization techniques

must be employed in the modem and data terminal equipment. Since the phase-shift-keying used in the ALOHANET modem design is a bit-synchronous technique, bit synchronization must first be performed in the demodulator before packet synchronization can be attempted. Bit-sync is performed by a phase-locking circuit, and a lock-indication signal is passed to the data equipment when bit-sync has been attained. The bit-sync detection circuit is so designed to provide a very low false detection probability (less than 10^{-8}) and a high probability of packet detection. The narrow bandwidth of the phase-lock circuit presently designed into the ALOHANET modem requires a bit-sync preamble of 90 bits to ensure reliable bit-sync. Studies have indicated that this preamble can be reduced to about 10 bits by use of a redesigned wide-band phase-lock circuit. In fact, we are presently contemplating doing away with the bit-sync preamble entirely, further reducing packet overhead. The unique characteristics of the ALOHA modem design make such an approach feasible.

Packet synchronization is accomplished in the ALOHANET data terminal buffer by means of the 16-bit parity word contained in the packet header. When the parity check routine accepts the header, the packet is assumed to be synchronized. Since the parity check routine is initiated by the first bit of the header, packets can be missed due to detection of an early error bit before the header. This miss probability is presently controlled by the modem at about 10^{-3} or less, providing a packet detection probability of 99.9 percent or better. The false detection probability of this circuit is $\sim 1.5\times10^{-8}$, which is independent of that of the modem. Thus, the overall probability of false detection is less than 1.5×10^{-11}. Therefore, less than one out of a thousand packets will be lost due to packet sync errors and packet sync false alarms occur with extreme rarity.

USER INTERFACE CHOICES

The development of the ALOHANET user interface has been an evolutionary process, as is typical of most research developments. Since there were expected to be many user nodes (as compared to the single MENEHUNE node), the primary design goals were initially set as simplicity of design and low cost. This led to the design of a hardwired control unit with limited data storage capability coupled to a modem and radio transceiver. This initial design was termed a Terminal Control Unit (TCU). As experience developed with operation of the net, other functions became evident as being desirable in a TCU. At about this time the first microprocessor chips and low-cost semiconductor memory chips were becoming available in the marketplace. It was decided that a new TCU design should be initiated using these new devices since much greater flexibility and additional functions could be readily incorporated in a unit having a capability of being programmed. It was also noted that the cost of these new devices was such that a unit could be built for the same cost or less than that of the original design. Thus, the Programmable Control Unit (PCU) was developed, and there are now several operating units in the system. We will now discuss some of the issues involved in designing a terminal control unit for use on the ALOHANET. These issues lie in the general areas of interface considerations, and the technology of microprocessors.

The original TCU

The ALOHANET was originally envisioned as a terminal network, with the TCU's interfacing human users to a half duplex, line-oriented time-sharing system. At the time of the first TCU design effort memory was relatively expensive, so in order to minimize cost a single buffer was chosen for use with both the terminal keyboard and display. (As noted earlier in this paper, when full duplex computer interactions were available in the system the single buffer was found to be quite a disadvantage.) The buffer was designed for a full line length of 80 characters, which allowed handling of both the 40 and 80 character fixed-length packets defined for the system.

Additional basic functions performed by the TCU's were generation of a cyclic-parity-check code vector and decoding of received parity code words for error-detection purposes, and generation of packet retransmissions using a simple random interval generator. If an acknowledgment was not received from the computer after the prescribed number of retransmissions, a flashing light was used as an indicator to the human user. Since the TCU's did not send acknowledgments to the MENEHUNE, a steady warning light was displayed to the human user when an error was detected in a received packet. Thus it can be seen that considerable simplification was incorporated into the initial design of the TCU, making use of the fact that it was interfacing a human user into the network.

Other functions hardwired into the TCU were the obvious requirements of checking for and generating its address, packet sequence numbering, checking to see if a received packet is an ACK packet or a data packet, and generating and checking for half- or full-packet conditions. (The control bits for these functions all reside in the header portion of the packet.)

The final consideration was the choice of standard interface signals between the TCU and the user's equipment. This was a relatively simple choice, since most equipment is designed to meet the EIA standard RS 232C interface specification. Therefore, the TCU was designed to meet this standard, which allows direct connection of most terminals in use today.

Minicomputer nodes

As the ALOHANET developed, some minicomputers were interfaced into the network as concentrators for a number of terminals. Many of the logical functions performed in a TCU were now incorporated into the mini's software, with error detection and parity word generation performed in a special hardware interface unit imposed between the minicomputer and an ALOHA modem. (This unit was very much like the encoder/decoder unit used at the MENEHUNE to interface that minicomputer to the channel.) Parallel-to-serial and serial-to-parallel conversion was also performed in this interface unit.

However, a minicomputer is an expensive device to use for these simple functions, and it requires considerable amounts of power and space. If it already exists for the purpose of performing various user-oriented tasks, then it is cost-effective to incorporate the software interface and a minimal amount of hardware for use on the ALOHANET.

The advent of the microprocessor chip changed all this. The relatively low-cost processing power demonstrated by these units made it apparent that many system options we had previously considered and discarded because of hardware complexity and cost limitations in the TCU, were now viable in a PCU. Some of these options—file transfer, remote user ACKs, single frequency operation, character-by-character transmission—were discussed in previous sections. This trend toward programmable and more powerful TCU's has thus led to the development of the ALOHA PCU, using a microprocessor to handle the TCU buffering and control functions, in addition to more complex and sophisticated functions.

Microprocessor technology

The development from the hardwired TCU concept to the fully-programmable PCU has closely followed the rapidly changing technology of microprocessors. The availability of lower-cost semiconductor memory has allowed the evolution from half-duplex to full-duplex operation in the PCU, with the beneficial side-effect of decreased logical complexity due to separation of the input and output functions. However, the first PCU developed had a hardware complexity level comparable to the TCU due to the relatively primitive structure of early microprocessor designs. This first PCU, designed with the Intel 8008 CPU, required a considerable amount of circuitry for buffering and multiplexing functions needed with this early microprocessor chip. Because of the slow speed of the chip, bit-by-bit processing was not possible and additional buffering was also necessary. But, much greater flexibility was introduced into the scope of functions which could be performed, due to its programmability.

Later microprocessor designs, such as the Intel 8080 and National IMP-16, have introduced much greater sophistication into the processor chips accompanied by significant processing speed improvements. A newer PCU design, incorporating an Intel 8080 chip, has demonstrated a considerable reduction in hardware complexity accompanied by an even greater degree of processing flexibility. For example, parity generation and checking are done in software with this prototype design.

Buffering has progressed from the simple shift-register storage devices of the TCU to the use of semiconductor RAM devices used in the microprocessor's random-access memory. All of the micro-instructions for the Intel 8080 microprocessor PCU design reside on four PROM chips, providing 1024 bytes of microcode. The random-access memory consists of 2048 bytes of RAM.

Recent product introductions such as Intel's 3000 series bi-polar chips promise even greater reductions in chip counts and increases in processing power and speed. With machines such as these, bit-by-bit processing can be readily incorporated into software, thus further eliminating the need for external interfacing hardware and simultaneously providing greater flexibility in the implementation of additional functions. A more detailed discussion of communications microprocessors is given in a companion paper in these proceedings.[9]

Size and power

In the earlier versions of the TCU smaller size and power drain of the unit were not considered major design objectives. The first units were designed for ease of access and hardware modifications to these TCU's were made on a fairly casual basis. As more and more of the ALOHANET came into use, however, small size, portability and lower power drain became desirable.

Of particular interest is the possibility of designing low power battery operated portable PCU's for mobile units in the ALOHANET. Since the transmitter power need only be on for a short burst corresponding to the period of the data burst, the average power of the transmitter can be a small percentage of the peak power. Since low power and small size were not original design objectives, it appears that the construction of low power portable PCU's will involve redesign of several subsections of the PCU and some new design efforts. Of particular importance is selection of a microprocessor unit which provides a minimum power-drain computer architecture consistent with functional requirements. The modem should be redesigned to use MOS devices to minimize power drain, and the transceiver designed for minimum complexity.

CONCLUSIONS

As the system has been modified during the past several years it has become apparent that packet broadcasting architecture is remarkably flexible in its tolerance of hardware, system and protocol modifications. This flexibility follows from the packet verification algorithms which lie at the basis of packet broadcasting. The only packets accepted by a remote unit or by the MENEHUNE are packets which meet all the tests expected by the potential acceptor; and the only system resource consumed by an unaccepted packet is the capacity of the channel during the short burst of the packet duration. Thus it is perfectly feasible in a packet broadcasting network to introduce a new form of packet (new in format, new in packet length, or even new in

modulation technique) without disturbing any unit operating with the existing scheme. Only the units designed to look for the new packets will accept these packets and all other units will simply discard them.

We plan to employ this property of packet switched channels to switch the polynomial used for error control in the present packet format. The new polynomial is available in a single IC chip and will allow the possibility of error correction as well as error detection in some cases. As remote units with new packet formats are put into operation we can continue to operate the existing remote units without modification as long as we have a single unit capable of accepting the new packet format at the MENEHUNE. As a side benefit of the introduction of this modification we also note that we have effectively doubled the number of user addresses in the system. An address in use with the old packet format may be reused with the new, since each is effectively invisible to the other.

Another result of our ALOHANET experience, current technology, and recent theoretical work on ALOHA channels, is that a *single-channel* network configuration appears preferable to the two channels used in our present system. The major reason why this is so has to do with the broadcast property of the single-channel system, in which all nodes can (for a given geographic range) hear the transmission of all other nodes in the net.

A number of desirable properties result from this broadcast feature. First, each node can determine if the channel is free before transmitting, greatly reducing the number of packet conflicts—Kleinrock and Tobagi[15] have shown analytically that this can increase the throughput of a random access channel by a factor of three to five for reasonable user delays, depending on the propagation times between nodes. Second, the problem of sending acknowledgments from user nodes is resolved in a simple manner. Third, system bandwidth can be optimally allocated to both directions of traffic by simple time-sharing of the channel. Fourth, single channel repeaters require only half the radio hardware of two-channel repeaters, and, in fact, the radio transceivers at all nodes need be only half duplex. Finally, a single-channel system constitutes a fully-connected network allowing direct communication between all nodes. A star configuration can still be imposed by protocol to direct all user traffic through a central node, but is no longer required.

It is important to note that many of the above properties are made feasible by the availability of PCU's at a reasonable cost through microcomputer technology. This raises a related issue: the desirability of distributing presently centralized protocol functions such as flow control among the user nodes. Since we have just begun to gain experience with PCU's in a packet broadcast network, we must leave this as an open question.

REFERENCES

1. Roberts, L. G. and B. Wessler, "The ARPA Computer Network," in *Computer Communication Networks* edited by Abramson and Kuo, Prentice-Hall, 1973.
2. Kahn, R. E., "The Organization of Computer Resources into a Packet Radio Network," in this issue, *AFIPS Conference Proceedings*, Volume 44.
3. Abramson, N., "Packet Switching with Satellites," *AFIPS Conference Proceedings*, Volume 42, 1973, pp. 695-702.
4. Binder, R., W. S. Lai and M. Wilson, *The ALOHANET MENEHUNE—Version II*, ALOHA SYSTEM Technical Report B74-6, University of Hawaii, September 1974.
5. Heart, F. E., R. E. Kahn, S. M. Ornstein, W. R. Crowther and D. C. Walden, "The Interface Message Processor for the ARPA Computer Network," *AFIPS Conference Proceedings*, Volume 36, 1970, pp. 543-567.
6. Ornstein, S. M., F. E. Heart, W. R. Crowther, H. K. Rising, S. B. Russell and A. Michel, "The Terminal IMP for the ARPA Computer Network," *AFIPS Conference Proceedings*, Volume 40, 1972, pp. 243-254.
7. Jackson, P. E. and C. D. Stubbs, "A Study of Multi-Access Computer Communications," *AFIPS Conference Proceedings*, Volume 34, 1969, pp. 491-504.
8. Abramson, N., "The ALOHA System," in *Computer-Communication Networks* edited by Abramson and Kuo, Prentice-Hall, 1973.
9. Fralick, S. C., D. H. Brandin, F. F. Kuo and C. Harrison, "Digital Terminals for Packet Broadcasting," in this issue, *AFIPS Conference Proceedings*, Volume 44.
10. Metcalfe, R. M., *Packet Communication*, Project MAC Technical Report, MAC TR-114, MIT, July 1973.
11. Lam, S. S., *Packet Switching in a Multi-Access Broadcast Channel with Application to Satellite Communication in a Computer Network*, UCLA-ENG-7429 Report, University of California at Los Angeles, Computer Science Department, April 1974.
12. Davidson, J., *An Echoing Strategy for Satellite Links*, NIC Document 10599, (RFC 357), Stanford Research Institute, June 1972.
13. Frank, H., R. M. Van Slyke and I. Gitman, "Packet Radio Network Design—System Considerations," in this issue, *AFIPS Conference Proceedings*, Volume 44.
14. Okumura, Y., E. Ohmori, T. Kawano and K. Fukuda, "Field Strength and its Variability in UHF and VHF Land-Mobile Radio Service," *Review of the Electrical Communication Laboratory*, Vol. 16, No. 9-10, September-October 1968.
15. Kleinrock, L. and F. Tobagi, "Packet Switching in Radio Channels: Carrier Sense Multiple-Access Modes and their Throughput Delay Characteristics," to be published in *IEEE Transactions on Communications*

Bugs in the Point to Point software

There are a few bugs in the point-to-point software which have been reported and are being worked on:

1. The first message after connection has been established will contain the call sign and the CTRL-X: VE3DVV(CTRL-X)Hi John... This is because the TIP must save all data entered up to the CTRL-X in case it is a general transmission. The buffer pointer is not reset after the connection is established.

2. When you type enough characters to the TIP to fill the available buffer space, and do not give a terminator (LF), the controller hangs. This is because the buffer is checked for FULL before the input character is examined to see if it is a terminator. If the buffer is full, the characters input are just ignored.

3. If the TIP backs up and the LIP buffer becomes full, RNR is sent back over the link, and the other station just keeps trying. It appears that in one or both of the stations, the N(R) or N(S) counters are messed up and the messages sent after the RR are rejected because they are out of sequence.

4. A zero length packet (no I field) (not possible to send with the distribution TIP) causes one or both stations to disconnect.

Further information will be published as it becomes available.